



本文档来自 **安卓巴士** ([www.apkbus.com](http://www.apkbus.com)) 整理总结

## 《Android 开发提高十六技》

### 作者简介:

---

张国威 (用户名: hellogv)

<http://hi.csdn.net/hellogv>

- 性别: 男
- 生日: 1985 年 11 月 20 日
- 婚恋: 单身
- 居住: 江苏 南京
- 家乡: 广东 湛江
- MSN: hellogv@qq.com
- Email: hellogv@QQ.com

(注: 本文档用于学习开发使用, 勿作为任何商业活动使用, 如作者有版权问题, 请联系安卓巴士网友: 巴士小白)

## 目录:

Android 提高第一篇之 MediaPlayer-----	3
Android 提高第二篇之 SurfaceView(上)-----	10
Android 提高第三篇之 SurfaceView(下)-----	18
Android 提高第四篇之 Activity+Intent-----	25
Android 提高第五篇之 Service-----	33
Android 提高第六篇之 BroadcastReceiver-----	41
Android 提高第七篇之 XML 解析与生成-----	48
Android 提高第八篇之 SQLite 分页读取-----	56
Android 提高第九篇之 SQLite 分页表格-----	64
Android 提高第十篇之 AudioRecord 实现"助听器"-----	75
Android 提高第十一篇之模拟信号示波器-----	81
Android 提高第十二篇之蓝牙传感应用-----	91
Android 提高第十三篇之探秘蓝牙隐藏 API-----	95
Android 提高第十四篇之探秘 TelephonyManager-----	105
Android 提高第十五篇之 ListView 自适应实现表格-----	113
Android 提高十六篇之使用 NDK 把彩图转换灰度图-----	120



本文档来自 **安卓巴士** ([www.apkbus.com](http://www.apkbus.com)) 整理总结

作者简介:

---

张国威 (用户名: hellogv)

<http://hi.csdn.net/hellogv>

- 性别: 男
- 生日: 1985 年 11 月 20 日
- 婚恋: 单身
- 居住: 江苏 南京
- 家乡: 广东 湛江
- MSN: hellogv@qq.com
- Email: hellogv@QQ.co

## Android 提高第一篇之 MediaPlayer

---

前面写了十四篇关于界面的入门文章,大家都看完和跟着练习之后,对于常用的 **Layout** 和 **View** 都会有一定的了解了,接下来的文章就不再强调介绍界面了,而是针对具体的常见功能而展开。

本文介绍 **MediaPlayer** 的使用。**MediaPlayer** 可以播放音频和视频,另外也可以通过 **VideoView** 来播放视频,虽然 **VideoView** 比 **MediaPlayer** 简单易用,但定制性不如用 **MediaPlayer**,要视情况选择了。**MediaPlayer** 播放音频比较简单,但是要播放视频就需要 **SurfaceView**。**SurfaceView** 比普通的自定义 **View** 更有绘图上的优势,它支持完全的 **OpenGL ES** 库。

先贴出本程序运行结果的截图，上面是播放/停止音频，可用 **SeekBar** 来调进度，下面是播放/停止视频，也是用 **SeekBar** 来调进度：



**main.xml** 的源码：

[view plaincopy to clipboardprint?](#)

```
1. <?xml version="1.0" encoding="utf-8"?>
2. <LinearLayout android:id="@+id/LinearLayout01"
3.     android:layout_width="fill_parent" android:layout_height="fill_parent"
4.     xmlns:android="http://schemas.android.com/apk/res/android"
5.     android:orientation="vertical">
6.     <SeekBar android:id="@+id/SeekBar01" android:layout_height="wrap_content"
7.         "
8.         android:layout_width="fill_parent"></SeekBar>
9.     <LinearLayout android:id="@+id/LinearLayout02"
```

```

9.         android:layout_width="wrap_content" android:layout_height="wrap_content">
10.         <Button android:id="@+id/Button01" android:layout_width="wrap_content"
11.             android:layout_height="wrap_content" android:text="播放音频"
12.             "></Button>
13.         <Button android:id="@+id/Button02" android:layout_width="wrap_content"
14.             android:layout_height="wrap_content" android:text="停止播放"
15.             "></Button>
16.     </LinearLayout>
17.     <SeekBar android:id="@+id/SeekBar02" android:layout_height="wrap_content"
18.         "
19.         android:layout_width="fill_parent"></SeekBar>
20.
21.     <SurfaceView android:id="@+id/SurfaceView01"
22.         android:layout_width="fill_parent" android:layout_height="250px"></SurfaceView>
23.     <LinearLayout android:id="@+id/LinearLayout02"
24.         android:layout_width="wrap_content" android:layout_height="wrap_content">
25.         <Button android:layout_width="wrap_content"
26.             android:layout_height="wrap_content" android:id="@+id/Button03"
27.             android:text="播放视频"></Button>
28.         <Button android:layout_width="wrap_content"
29.             android:layout_height="wrap_content" android:text="停止播放"
30.             " android:id="@+id/Button04"></Button>
31.     </LinearLayout>
32. </LinearLayout>

```

本文程序的源码，有点长：

[view plaincopy to clipboardprint?](#)

```

1. package com.testMedia;
2.
3. import java.io.IOException;
4. import java.util.Timer;
5. import java.util.TimerTask;
6. import android.app.Activity;
7. import android.media.AudioManager;

```

```

8. import android.media.MediaPlayer;
9. import android.os.Bundle;
10. import android.view.SurfaceHolder;
11. import android.view.SurfaceView;
12. import android.view.View;
13. import android.widget.Button;
14. import android.widget.SeekBar;
15. import android.widget.Toast;
16.
17.
18. public class testMedia extends Activity {
19.     /** Called when the activity is first created. */
20.
21.     private SeekBar skb_audio=null;
22.     private Button btn_start_audio = null;
23.     private Button btn_stop_audio = null;
24.
25.     private SeekBar skb_video=null;
26.     private Button btn_start_video = null;
27.     private Button btn_stop_video = null;
28.     private SurfaceView surfaceView;
29.     private SurfaceHolder surfaceHolder;
30.
31.     private MediaPlayer m = null;
32.     private Timer mTimer;
33.     private TimerTask mTimerTask;
34.
35.     private boolean isChanging=false;//互斥变量，防止定时器与 SeekBar 拖动时进度
        冲突
36.     @Override
37.     public void onCreate(Bundle savedInstanceState) {
38.         super.onCreate(savedInstanceState);
39.         setContentView(R.layout.main);
40.
41.         //-----Media 控件设置-----//
42.         m=new MediaPlayer();
43.
44.         //播放结束之后弹出提示
45.         m.setOnCompletionListener(new MediaPlayer.OnCompletionListener(){
46.             @Override
47.             public void onCompletion(MediaPlayer arg0) {
48.                 Toast.makeText(testMedia.this, "结束", 1000).show();
49.                 m.release();
50.             }

```

```

51.         });
52.
53.         //-----定时器记录播放进度-----//
54.         mTimer = new Timer();
55.         mTimerTask = new TimerTask() {
56.             @Override
57.             public void run() {
58.                 if(isChanging==true)
59.                     return;
60.
61.                 if(m.getVideoHeight()==0)
62.                     skb_audio.setProgress(m.getCurrentPosition());
63.                 else
64.                     skb_video.setProgress(m.getCurrentPosition());
65.             }
66.         };
67.
68.         mTimer.schedule(mTimerTask, 0, 10);
69.
70.         btn_start_audio = (Button) this.findViewById(R.id.Button01);
71.         btn_stop_audio = (Button) this.findViewById(R.id.Button02);
72.         btn_start_audio.setOnClickListener(new ClickEvent());
73.         btn_stop_audio.setOnClickListener(new ClickEvent());
74.         skb_audio=(SeekBar)this.findViewById(R.id.SeekBar01);
75.         skb_audio.setOnSeekBarChangeListener(new SeekBarChangeListener());
76.
77.         btn_start_video = (Button) this.findViewById(R.id.Button03);
78.         btn_stop_video = (Button) this.findViewById(R.id.Button04);
79.         btn_start_video.setOnClickListener(new ClickEvent());
80.         btn_stop_video.setOnClickListener(new ClickEvent());
81.         skb_video=(SeekBar)this.findViewById(R.id.SeekBar02);
82.         skb_video.setOnSeekBarChangeListener(new SeekBarChangeListener());
83.         surfaceView = (SurfaceView) findViewById(R.id.SurfaceView01);
84.         surfaceHolder = surfaceView.getHolder();
85.         surfaceHolder.setFixedSize(100, 100);
86.         surfaceHolder.setType(SurfaceHolder.SURFACE_TYPE_PUSH_BUFFERS);
87.     }
88.
89.     /*
90.      * 按键事件处理
91.      */
92.     class ClickEvent implements View.OnClickListener{
93.         @Override
94.         public void onClick(View v) {

```

```

95.         if(v==btn_start_audio)
96.         {
97.             m.reset();//恢复到未初始化的状态
98.             m=MediaPlayer.create(testMedia.this, R.raw.big);//读取音频
99.             skb_audio.setMax(m.getDuration());//设置 SeekBar 的长度
100.            try {
101.                m.prepare();    //准备
102.            } catch (IllegalStateException e) {
103.                // TODO Auto-generated catch block
104.                e.printStackTrace();
105.            } catch (IOException e) {
106.                // TODO Auto-generated catch block
107.                e.printStackTrace();
108.            }
109.            m.start(); //播放
110.        }
111.        else if(v==btn_stop_audio || v==btn_stop_video)
112.        {
113.            m.stop();
114.        }
115.        else if(v==btn_start_video)
116.        {
117.            m.reset();//恢复到未初始化的状态
118.            m=MediaPlayer.create(testMedia.this, R.raw.test);//读取视频
119.            skb_video.setMax(m.getDuration());//设置 SeekBar 的长度
120.            m.setAudioStreamType(AudioManager.STREAM_MUSIC);
121.            m.setDisplay(surfaceHolder);//设置屏幕
122.
123.            try {
124.                m.prepare();
125.
126.            } catch (IllegalArgumentException e) {
127.                // TODO Auto-generated catch block
128.                e.printStackTrace();
129.            } catch (IllegalStateException e) {
130.                // TODO Auto-generated catch block
131.                e.printStackTrace();
132.            } catch (IOException e) {
133.                // TODO Auto-generated catch block
134.                e.printStackTrace();
135.            }
136.            m.start();
137.        }
138.    }

```



```
139.     }
140.
141.     /*
142.      * SeekBar 进度改变事件
143.      */
144.     class SeekBarChangeEvent implements SeekBar.OnSeekBarChangeListener{
145.
146.         @Override
147.         public void onProgressChanged(SeekBar seekBar, int progress,
148.             boolean fromUser) {
149.             // TODO Auto-generated method stub
150.
151.         }
152.
153.         @Override
154.         public void onStartTrackingTouch(SeekBar seekBar) {
155.             isChanging=true;
156.         }
157.
158.         @Override
159.         public void onStopTrackingTouch(SeekBar seekBar) {
160.             m.seekTo(seekBar.getProgress());
161.             isChanging=false;
162.         }
163.
164.     }
165.
166. }
```



本文档来自 **安卓巴士** ([www.apkbus.com](http://www.apkbus.com)) 整理总结

## 作者简介:

---

张国威 (用户名: hellogv)

<http://hi.csdn.net/hellogv>

- 性别: 男
- 生日: 1985 年 11 月 20 日
- 婚恋: 单身
- 居住: 江苏 南京
- 家乡: 广东 湛江
- MSN: hellogv@qq.com
- Email: [hellogv@QQ.com](mailto:hellogv@QQ.com)
- 

## Android提高第二篇之SurfaceView(上)

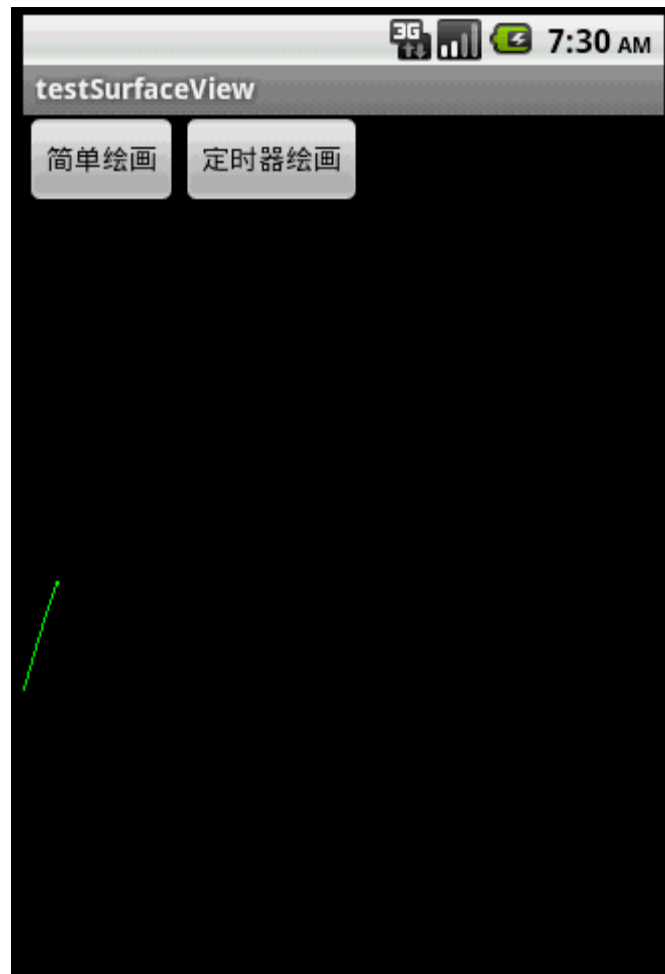
---

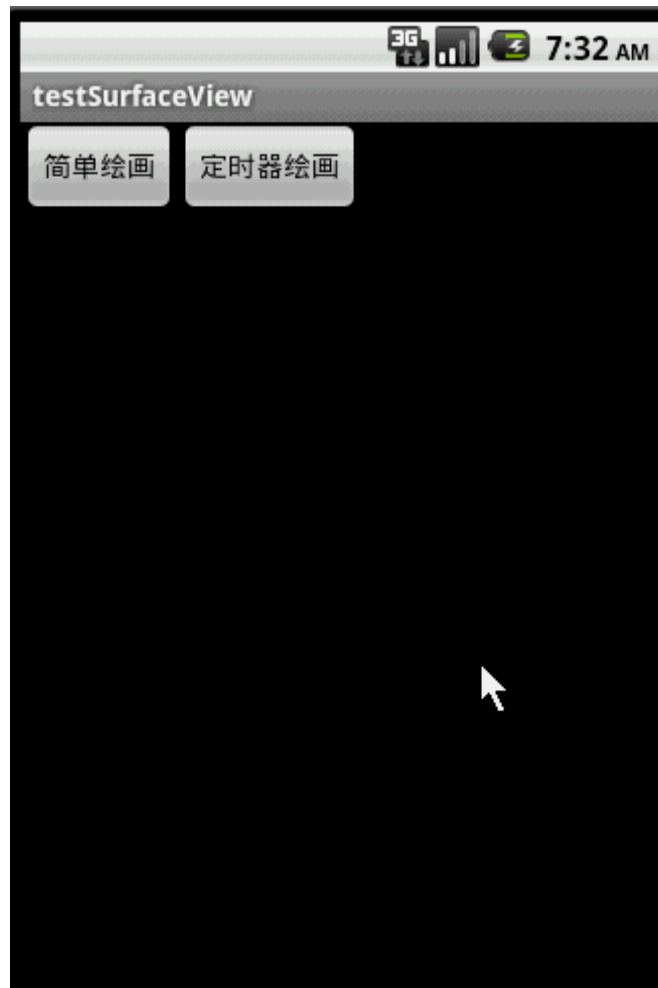
上次介绍MediaPlayer的时候稍微介绍了SurfaceView, SurfaceView由于可以直接从内存或者DMA等硬件接口取得图像数据, 因此是个非常重要的绘图容器, 这次我就用两篇文章来介绍SurfaceView的用法。网上介绍SurfaceView的用法有很多, 写法也层出不同, 例如继承SurfaceView类, 或者继承SurfaceHolder.Callback类等, 这个可以根据功能实际需要自己选择, 我这里就直接在普通的用户界面调用SurfaceHolder的lockCanvas和unlockCanvasAndPost。

先来看看程序运行的截图:



截图 1 主要演示了直接把正弦波绘画在 `SurfaceView` 上





对比上面的左右两图，右图用`.lockCanvas(null)`，而左图用`.lockCanvas(new Rect(oldX, 0, oldX + length, getWindowManager().getDefaultDisplay().getHeight()))`，对比一下两个效果，由于左图是按指定 `Rect` 绘画，所以效率会比右图的全控件绘画高些，并且在清屏之后(`canvas.drawColor(Color.BLACK)`)不会留有上次绘画的残留。

接下来贴出 `main.xml` 的源码:

[view plaincopy to clipboardprint?](#)

```
1. <?xml version="1.0" encoding="utf-8"?>
```

```

2. <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
3.     android:layout_width="fill_parent" android:layout_height="fill_parent"
4.     android:orientation="vertical">
5.
6.     <LinearLayout android:id="@+id/LinearLayout01"
7.         android:layout_width="wrap_content" android:layout_height="wrap_cont
8.         ent">
9.         <Button android:id="@+id/Button01" android:layout_width="wrap_conten
10.        t"
11.            android:layout_height="wrap_content" android:text="简单绘画
12.        "></Button>
13.        <Button android:id="@+id/Button02" android:layout_width="wrap_conten
14.        t"
15.            android:layout_height="wrap_content" android:text="定时器绘画
16.        "></Button>
17.    </LinearLayout>
18.    <SurfaceView android:id="@+id/SurfaceView01"
19.        android:layout_width="fill_parent" android:layout_height="fill_paren
20.        t"></SurfaceView>
21. </LinearLayout>

```

接下来贴出程序源码：

[view plaincopy to clipboardprint?](#)

```

1. package com.testSurfaceView;
2.
3. import java.util.Timer;
4. import java.util.TimerTask;
5.
6. import android.app.Activity;
7. import android.graphics.Canvas;
8. import android.graphics.Color;
9. import android.graphics.Paint;
10. import android.graphics.Rect;
11. import android.os.Bundle;
12. import android.util.Log;
13. import android.view.SurfaceHolder;
14. import android.view.SurfaceView;
15. import android.view.View;
16. import android.widget.Button;
17.
18. public class testSurfaceView extends Activity {
19.     /** Called when the activity is first created. */

```

```

20.     Button btnSimpleDraw, btnTimerDraw;
21.     SurfaceView sfv;
22.     SurfaceHolder sfh;
23.
24.     private Timer mTimer;
25.     private MyTimerTask mTimerTask;
26.     int Y_axis[],//保存正弦波的 Y 轴上的点
27.     centerY,//中心线
28.     oldX,oldY,//上一个 XY 点
29.     currentX;//当前绘制到的 X 轴上的点
30.
31.     @Override
32.     public void onCreate(Bundle savedInstanceState) {
33.         super.onCreate(savedInstanceState);
34.         setContentView(R.layout.main);
35.
36.         btnSimpleDraw = (Button) this.findViewById(R.id.Button01);
37.         btnTimerDraw = (Button) this.findViewById(R.id.Button02);
38.         btnSimpleDraw.setOnClickListener(new ClickEvent());
39.         btnTimerDraw.setOnClickListener(new ClickEvent());
40.         sfv = (SurfaceView) this.findViewById(R.id.SurfaceView01);
41.         sfh = sfv.getHolder();
42.
43.         //动态绘制正弦波的定时器
44.         mTimer = new Timer();
45.         mTimerTask = new MyTimerTask();
46.
47.         // 初始化 y 轴数据
48.         centerY = (getWindowManager().getDefaultDisplay().getHeight() - sfv
49.             .getTop()) / 2;
50.         Y_axis = new int[getWindowManager().getDefaultDisplay().getWidth()];
51.         for (int i = 1; i < Y_axis.length; i++) {// 计算正弦波
52.             Y_axis[i - 1] = centerY
53.                 - (int) (100 * Math.sin(i * 2 * Math.PI / 180));
54.         }
55.     }
56.
57.     class ClickEvent implements View.OnClickListener {
58.
59.         @Override
60.         public void onClick(View v) {
61.

```

```

62.         if (v == btnSimpleDraw) {
63.             SimpleDraw(Y_axis.length-1); //直接绘制正弦波
64.
65.         } else if (v == btnTimerDraw) {
66.             oldY = centerY;
67.             mTimer.schedule(mTimerTask, 0, 5); //动态绘制正弦波
68.         }
69.
70.     }
71.
72. }
73.
74. class MyTimerTask extends TimerTask {
75.     @Override
76.     public void run() {
77.
78.         SimpleDraw(currentX);
79.         currentX++; //往前进
80.         if (currentX == Y_axis.length - 1) { //如果到了终点，则清屏重来
81.             ClearDraw();
82.             currentX = 0;
83.             oldY = centerY;
84.         }
85.     }
86.
87. }
88.
89. /*
90.  * 绘制指定区域
91.  */
92. void SimpleDraw(int length) {
93.     if (length == 0)
94.         oldX = 0;
95.     Canvas canvas = sfh.lockCanvas(new Rect(oldX, 0, oldX + length,
96.         getWindowManager().getDefaultDisplay().getHeight())); // 关键：
97.     Log.i("Canvas:",
98.         String.valueOf(oldX) + "," + String.valueOf(oldX + length));
99.
100.     Paint mPaint = new Paint();
101.     mPaint.setColor(Color.GREEN); // 画笔为绿色
102.     mPaint.setStrokeWidth(2); // 设置画笔粗细
103.

```



```

104.         int y;
105.         for (int i = oldX + 1; i < length; i++) { // 绘画正弦波
106.             y = Y_axis[i - 1];
107.             canvas.drawLine(oldX, oldY, i, y, mPaint);
108.             oldX = i;
109.             oldY = y;
110.         }
111.         sfh.unlockCanvasAndPost(canvas); // 解锁画布, 提交画好的图像
112.     }
113.
114.     void ClearDraw() {
115.         Canvas canvas = sfh.lockCanvas(null);
116.         canvas.drawColor(Color.BLACK); // 清除画布
117.         sfh.unlockCanvasAndPost(canvas);
118.
119.     }
120. }

```

注意一下 `for (int i = oldX + 1; i < length; i++) { // 绘画正弦波` 这句，在 `.lockCanvas()` 指定 `Rect` 内减少循环画线的次数，可以提高绘图效率。



本文档来自 **安卓巴士** ([www.apkbus.com](http://www.apkbus.com)) 整理总结

## 作者简介:

---

张国威 (用户名: hellogv)

<http://hi.csdn.net/hellogv>

- 性别: 男
- 生日: 1985 年 11 月 20 日
- 婚恋: 单身
- 居住: 江苏 南京
- 家乡: 广东 湛江
- MSN: hellogv@qq.com
- Email: [hellogv@QQ.com](mailto:hellogv@QQ.com)
- 

## Android提高第三篇之SurfaceView(下)

---

[上一篇](#)简单介绍了**SurfaceView**的使用,这次就介绍**SurfaceView**的双缓冲使用。双缓冲是为了防止动画闪烁而实现的一种多线程应用,基于**SurfaceView**的双缓冲实现很简单,开一条线程并在其中绘图即可。本文介绍基于**SurfaceView**的双缓冲实现,以及介绍类似的更高效的实现方法。

本文程序运行截图如下,左边是开单个线程读取并绘图,右边是开两个线程,一个专门读取图片,一个专门绘图:



对比一下，右边动画的帧速明显比左边的快，左右两者都没使用 `Thread.sleep()`。为什么要开两个线程一个读一个画，而不去开两个线程像左边那样都“边读边画”呢？因为 `SurfaceView` 每次绘图都会锁定 `Canvas`，也就是说同一片区域这次没画完下次就不能画，因此要提高双缓冲的效率，就得开一条线程专门画图，开另外一条线程做预处理的工作。

`main.xml` 的源码：

[view plaincopy to clipboardprint?](#)

```

1.  <?xml version="1.0" encoding="utf-8"?>
2.  <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
3.      android:layout_width="fill_parent" android:layout_height="fill_parent"
4.      android:orientation="vertical">
5.
6.      <LinearLayout android:id="@+id/LinearLayout01"
7.          android:layout_width="wrap_content" android:layout_height="wrap_content">
8.          <Button android:id="@+id/Button01" android:layout_width="wrap_content"
9.              android:layout_height="wrap_content" android:text="单个独立线程"
10.          ></Button>
11.          <Button android:id="@+id/Button02" android:layout_width="wrap_content"
12.              android:layout_height="wrap_content" android:text="两个独立线程"
13.          ></Button>

```

```

12.     </LinearLayout>
13.     <SurfaceView android:id="@+id/SurfaceView01"
14.         android:layout_width="fill_parent" android:layout_height="fill_paren
        t"></SurfaceView>
15. </LinearLayout>

```

本文程序的源码:

[view plaincopy to clipboardprint?](#)

```

1. package com.testSurfaceView;
2.
3. import java.lang.reflect.Field;
4. import java.util.ArrayList;
5. import android.app.Activity;
6. import android.graphics.Bitmap;
7. import android.graphics.BitmapFactory;
8. import android.graphics.Canvas;
9. import android.graphics.Paint;
10. import android.graphics.Rect;
11. import android.os.Bundle;
12. import android.util.Log;
13. import android.view.SurfaceHolder;
14. import android.view.SurfaceView;
15. import android.view.View;
16. import android.widget.Button;
17.
18. public class testSurfaceView extends Activity {
19.     /** Called when the activity is first created. */
20.     Button btnSingleThread, btnDoubleThread;
21.     SurfaceView sfv;
22.     SurfaceHolder sfh;
23.     ArrayList<Integer> imgList = new ArrayList<Integer>();
24.     int imgWidth, imgHeight;
25.     Bitmap bitmap; //独立线程读取, 独立线程绘图
26.
27.     @Override
28.     public void onCreate(Bundle savedInstanceState) {
29.         super.onCreate(savedInstanceState);
30.         setContentView(R.layout.main);
31.

```

```

32.         btnSingleThread = (Button) this.findViewById(R.id.Button01);
33.         btnDoubleThread = (Button) this.findViewById(R.id.Button02);
34.         btnSingleThread.setOnClickListener(new ClickEvent());
35.         btnDoubleThread.setOnClickListener(new ClickEvent());
36.         sfv = (SurfaceView) this.findViewById(R.id.SurfaceView01);
37.         sfh = sfv.getHolder();
38.         sfh.addCallback(new MyCallBack()); // 自动运行 surfaceCreated 以及
        surfaceChanged
39.     }
40.
41.     class ClickEvent implements View.OnClickListener {
42.
43.         @Override
44.         public void onClick(View v) {
45.
46.             if (v == btnSingleThread) {
47.                 new Load_DrawImage(0, 0).start(); // 开一条线程读取并绘图
48.             } else if (v == btnDoubleThread) {
49.                 new LoadImage().start(); // 开一条线程读取
50.                 new DrawImage(imgWidth + 10, 0).start(); // 开一条线程绘图
51.             }
52.
53.         }
54.
55.     }
56.
57.     class MyCallBack implements SurfaceHolder.Callback {
58.
59.         @Override
60.         public void surfaceChanged(SurfaceHolder holder, int format, int width,
        int height) {
61.             Log.i("Surface:", "Change");
62.
63.         }
64.
65.
66.         @Override
67.         public void surfaceCreated(SurfaceHolder holder) {
68.             Log.i("Surface:", "Create");
69.
70.             // 用反射机制来获取资源中的图片 ID 和尺寸
71.             Field[] fields = R.drawable.class.getDeclaredFields();
72.             for (Field field : fields) {
73.                 if (!"icon".equals(field.getName())) // 除了 icon 之外的图片

```

```

74.         {
75.             int index = 0;
76.             try {
77.                 index = field.getInt(R.drawable.class);
78.             } catch (IllegalArgumentException e) {
79.                 // TODO Auto-generated catch block
80.                 e.printStackTrace();
81.             } catch (IllegalAccessException e) {
82.                 // TODO Auto-generated catch block
83.                 e.printStackTrace();
84.             }
85.             // 保存图片 ID
86.             imgList.add(index);
87.         }
88.     }
89.     // 取得图像大小
90.     Bitmap bmImg = BitmapFactory.decodeResource(getResources(),
91.         imgList.get(0));
92.     imgWidth = bmImg.getWidth();
93.     imgHeight = bmImg.getHeight();
94. }
95.
96. @Override
97. public void surfaceDestroyed(SurfaceHolder holder) {
98.     Log.i("Surface:", "Destroy");
99.
100. }
101.
102. }
103.
104. /*
105.  * 读取并显示图片的线程
106.  */
107. class Load_DrawImage extends Thread {
108.     int x, y;
109.     int imgIndex = 0;
110.
111.     public Load_DrawImage(int x, int y) {
112.         this.x = x;
113.         this.y = y;
114.     }
115.
116.     public void run() {
117.         while (true) {

```

```

118.         Canvas c = sfh.lockCanvas(new Rect(this.x, this.y, this.x
119.             + imgWidth, this.y + imgHeight));
120.         Bitmap bmImg = BitmapFactory.decodeResource(getResources(),
121.             imgList.get(imgIndex));
122.         c.drawBitmap(bmImg, this.x, this.y, new Paint());
123.         imgIndex++;
124.         if (imgIndex == imgList.size())
125.             imgIndex = 0;
126.
127.         sfh.unlockCanvasAndPost(c);// 更新屏幕显示内容
128.     }
129. }
130. };
131.
132. /*
133.  * 只负责绘图的线程
134.  */
135. class DrawImage extends Thread {
136.     int x, y;
137.
138.     public DrawImage(int x, int y) {
139.         this.x = x;
140.         this.y = y;
141.     }
142.
143.     public void run() {
144.         while (true) {
145.             if (bitmap != null) {//如果图像有效
146.                 Canvas c = sfh.lockCanvas(new Rect(this.x, this.y, this
147.                     .x
148.                     + imgWidth, this.y + imgHeight));
149.                 c.drawBitmap(bitmap, this.x, this.y, new Paint());
150.
151.                 sfh.unlockCanvasAndPost(c);// 更新屏幕显示内容
152.             }
153.         }
154.     }
155. };
156.
157. /*
158.  * 只负责读取图片的线程
159.  */

```

```
160.     class LoadImage extends Thread {
161.         int imgIndex = 0;
162.
163.         public void run() {
164.             while (true) {
165.                 bitmap = BitmapFactory.decodeResource(getResources(),
166.                     imgList.get(imgIndex));
167.                 imgIndex++;
168.                 if (imgIndex == imgList.size())//如果到尽头则重新读取
169.                     imgIndex = 0;
170.             }
171.         }
172.     };
173. }
```





本文档来自 **安卓巴士** ([www.apkbus.com](http://www.apkbus.com)) 整理总结

## 作者简介:

---

张国威 (用户名: hellogv)

<http://hi.csdn.net/hellogv>

- 性别: 男
- 生日: 1985 年 11 月 20 日
- 婚恋: 单身
- 居住: 江苏 南京
- 家乡: 广东 湛江
- MSN: [hellogv@qq.com](mailto:hellogv@qq.com)
- Email: [hellogv@QQ.com](mailto:hellogv@QQ.com)
- 

## Android提高第四篇之Activity+Intent

---

Android 有三个基础组件 **Activity**, **Service** 和 **BroadcastReceiver**, 他们都是依赖 **Intent** 来启动。本文介绍的是 **Activity** 的生命周期以及针对 **Activity** 的 **Intent** 使用。

之前的例子一直都是使用 **Activity**, 在一个 **Layout XML** 与一个 **Activity** 捆绑的情况下可以视为一个 **Form**, 多个 **Layout XML** 与一个 **Activity** 捆绑的话那就是个 **Application** 本身了。**Intent** 可以分为显式 **Intent** 和隐式 **Intent**: 显式 **Intent** 用于启动明确的目标组件(前面所说的三大组件), 同一个 **Application** 内的多个 **Activity** 调用也是显式 **Intent**; 隐式 **Intent** 就是调用没有明确的目标组件, 可以是系统也可

以是第三方程序。隐式 **Intent** 一般用于调用系统组件功能，相关例程都是网络上很容易找到的（调用某些系统组件的时候要申请权限）。

**Activity** 的运行状况分为：**onCreate**、**onDestroy**、**onStart**、**onStop**、**onRestart**、**onResume**、**onPause**，**onCreate** 对应 **onDestroy**，**onStart** 对应 **onStop**，**onResume** 对应 **onPause**。

先贴出本文运行截图：



Log		
pid	tag	Message
870	Activity1	onPause
870	Activity2	onCreate
870	Activity2	onStart
870	Activity2	onResume
870	Activity1	onStop
870	Activity1	onDestroy

这个是从 **Activity1** 转到 **Activity2** 的时候，**Activity1** 的状态变化，使用了 **finish()** 会触发 **onDestroy()**。

Log		
pid	tag	Message
870	Activity2	onPause
870	Activity1	onCreate
870	Activity1	onStart
870	Activity1	onResume
870	Activity2	onStop
870	Activity2	onDestroy

这个是从 **Activity2** 转到 **Activity1** 的时候，**Activity2** 的状态变化。

从两次 **Activity** 的启动可以看出，流程是

**onCreate()**->**onStart()**->**onResume()**三个方法，销毁是

**onPause()**->**onStop()**->**onDestroy()**。另外，要往工程添加第二个

**Activity**，需要到 **AndroidManifest.xml**->**Application** 那里添加

**Activity2**。

**main1.xml** 的源码:

[view plaincopy to clipboardprint?](#)

```

1. <?xml version="1.0" encoding="utf-8"?>
2. <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
3.     android:orientation="vertical" android:layout_width="fill_parent"
4.     android:layout_height="fill_parent">
5.     <Button android:layout_width="wrap_content"
6.         android:layout_height="wrap_content" android:id="@+id/main1.Button01"
7.         android:text="跳转到 Activity2"></Button>
8.     <EditText android:text="@+id/EditText01" android:id="@+id/EditText01"
9.         android:layout_width="wrap_content" android:layout_height="wrap_cont
ent"></EditText>
10.    <Button android:layout_width="wrap_content"
11.        android:layout_height="wrap_content" android:id="@+id/main1.Button02"
12.        android:text="跳转到外部 Activity"></Button>
13. </LinearLayout>

```

## main2.xml 的源码:

[view plain](#)[copy](#) to [clipboard](#)[print](#)?

```
1. <?xml version="1.0" encoding="UTF-8"?>
2. <LinearLayout android:id="@+id/LinearLayout01"
3.     android:layout_width="fill_parent" android:layout_height="fill_parent"
4.     xmlns:android="http://schemas.android.com/apk/res/android">
5.     <Button android:layout_width="wrap_content"
6.         android:layout_height="wrap_content" android:id="@+id/main2.Button01"
7.         android:text="返回 Activity1"></Button>
8. </LinearLayout>
```

## Activity1 的源码:

[view plain](#)[copy](#) to [clipboard](#)[print](#)?

```
1. package com.testActivityIntent;
2. import android.app.Activity;
3. import android.content.Intent;
4. import android.content.SharedPreferences;
5. import android.net.Uri;
6. import android.os.Bundle;
7. import android.util.Log;
8. import android.view.View;
9. import android.widget.Button;
10. import android.widget.EditText;
11. public class testActivityIntent extends Activity {
12.     /** Called when the activity is first created. */
13.     Button btnToInternalActivity;
14.     Button btnToExternalActivity;
15.     EditText tbBundle;
16.     @Override
17.     public void onCreate(Bundle savedInstanceState) {
18.         super.onCreate(savedInstanceState);
19.         Log.e("Activity1", "onCreate");//显示当前状态, onCreate 与 onDestroy 对应
20.         setContentView(R.layout.main1);
21.
22.         btnToInternalActivity=(Button)this.findViewById(R.id.main1_Button01)
23.         ;
24.         btnToExternalActivity=(Button)this.findViewById(R.id.main1_Button02)
25.         ;
```

```

24.         btnToInternalActivity.setOnClickListener(new ClickEvent());
25.         btnToExternalActivity.setOnClickListener(new ClickEvent());
26.         tbBundle=(EditText)this.findViewById(R.id.EditText01);
27.     }
28.     public void onDestroy()
29.     {
30.         super.onDestroy();
31.         Log.e("Activity1", "onDestroy");//显示当前状态, onCreate 与 onDestroy 对
    应
32.     }
33.     @Override
34.     public void onStart()
35.     {
36.         super.onStart();
37.         Log.e("Activity1", "onStart");//显示当前状态, onStart 与 onStop 对应
38.     }
39.     @Override
40.     public void onStop()
41.     {
42.         super.onStop();
43.         Log.e("Activity1", "onStop");//显示当前状态, onStart 与 onStop 对应
44.     }
45.     @Override
46.     public void onRestart()
47.     {
48.         super.onRestart();
49.         Log.e("Activity1", "onRestart");
50.     }
51.     @Override
52.     public void onResume()
53.     {
54.         super.onResume();
55.         Log.e("Activity1", "onResume");//显示当前状态, onPause 与 onResume 对
    应
56.         SharedPreferences prefs = getPreferences(0); //SharedPreferences 用于
    存储数据
57.         String restoredText = prefs.getString("editText01", null);
58.         if (restoredText != null) {
59.             this.tbBundle.setText(restoredText);
60.         }
61.     }
62.     @Override
63.     public void onPause()
64.     {

```

```

65.         super.onResume();
66.         Log.e("Activity1", "onPause");//显示当前状态, onPause 与 onResume 对
        应
67.         //保存文本框的内容, 使得重回本 Acitivity 的时候可以恢复
68.         SharedPreferences.Editor editor = getPreferences(0).edit();//SharedP
        references 用于存储数据
69.         editor.putString("editText01", this.tbBundle.getText().toString());
70.         editor.commit();
71.     }
72.
73.     class ClickEvent implements View.OnClickListener{
74.         @Override
75.         public void onClick(View v) {
76.             if(v==btnToInternalActivity)
77.             {
78.                 Intent intent = new Intent();
79.                 intent.setClass(testActivityIntent.this,Activity2.class);
80.
81.                 //new 一个 Bundle 对象, 并将要传递的数据传入
82.                 Bundle bundle = new Bundle();
83.                 bundle.putString("Text",tbBundle.getText().toString());
84.
85.                 //将 Bundle 对象 assign 给 Intent
86.                 intent.putExtras(bundle);
87.
88.                 //调用 Activity2
89.                 startActivity(intent);
90.
91.                 testActivityIntent.this.finish();//会触发 onDestroy();
92.             }
93.             else if(v==btnToExternalActivity)
94.             {
95.                 //有些外部调用需要开启权限
96.                 Uri uri = Uri.parse("http://google.com");
97.                 Intent it = new Intent(Intent.ACTION_VIEW, uri);
98.                 startActivity(it);
99.             }
100.
101.         }
102.
103.     }
104.
105. }

```

## Activity2 的源码:

[view plain](#)[copy to clipboard](#)[print?](#)

```
1. package com.testActivityIntent;
2. import android.app.Activity;
3. import android.content.Intent;
4. import android.os.Bundle;
5. import android.util.Log;
6. import android.view.View;
7. import android.widget.Button;
8. public class Activity2 extends Activity {
9.     Button btnBackMain1;
10.    public void onCreate(Bundle savedInstanceState)
11.    {
12.        super.onCreate(savedInstanceState);
13.        Log.e("Activity2", "onCreate");//显示当前状态, onCreate 与 onDestroy 对
        应
14.
15.        //加载 activity2.xml
16.        setContentView(R.layout.main2);
17.
18.        //得 Intent 中的 Bundle 对象
19.        Bundle bundle = this getIntent().getExtras();
20.        //取得 Bundle 对象中的数据
21.        Log.i("In_Text", bundle.getString("Text"));
22.        btnBackMain1=(Button)this.findViewById(R.id.main2_Button01);
23.        btnBackMain1.setOnClickListener(new ClickEvent());
24.    }
25.
26.    public void onDestroy()
27.    {
28.        super.onDestroy();
29.        Log.e("Activity2", "onDestroy");//显示当前状态, onCreate 与 onDestroy 对
        应
30.    }
31.    @Override
32.    public void onStart()
33.    {
34.        super.onStart();
35.        Log.e("Activity2", "onStart");//显示当前状态, onStart 与 onStop 对应
36.    }
37.    @Override
38.    public void onStop()
```

```

39.     {
40.         super.onStop();
41.         Log.e("Activity2", "onStop");//显示当前状态, onStart 与 onStop 对应
42.     }
43.     @Override
44.     public void onRestart()
45.     {
46.         super.onRestart();
47.         Log.e("Activity2", "onRestart");
48.     }
49.     @Override
50.     public void onResume()
51.     {
52.         super.onResume();
53.         Log.e("Activity2", "onResume");//显示当前状态, onPause 与 onResume 对
    应
54.     }
55.     @Override
56.     public void onPause()
57.     {
58.         super.onResume();
59.         Log.e("Activity2", "onPause");//显示当前状态, onPause 与 onResume 对
    应
60.     }
61.
62.     class ClickEvent implements View.OnClickListener{
63.         @Override
64.         public void onClick(View v) {
65.             if(v==btnBackMain1)
66.             {
67.
68.                 Intent intent = new Intent();
69.                 intent.setClass(Activity2.this,testActivityIntent.class);
70.
71.                 //调用 Activity1
72.                 startActivity(intent);
73.
74.                 Activity2.this.finish();//会触发 onDestroy();
75.             }
76.
77.         }
78.
79.     }
80. }

```





本文档来自 **安卓巴士** ([www.apkbus.com](http://www.apkbus.com)) 整理总结

## 作者简介:

---

张国威 (用户名: hellogv)

<http://hi.csdn.net/hellogv>

- 性别: 男
- 生日: 1985 年 11 月 20 日
- 婚恋: 单身
- 居住: 江苏 南京
- 家乡: 广东 湛江
- MSN: hellogv@qq.com
- Email: hellogv@QQ.com

## Android提高第五篇之Service

---

上次介绍了 [Activity](#) 以及 [Intent](#) 的使用, 这次就介绍 **Service**, 如果把 **Activity** 比喻为前台程序, 那么 **Service** 就是后台程序, **Service** 的整个生命周期都只会在后台执行。 **Service** 跟 **Activity** 一样也由 **Intent** 调用。在工程里想要添加一个 **Service**, 先新建继承 **Service** 的类, 然后到 **AndroidManifest.xml** -> **Application** -> **Application Nodes** 中的 **Service** 标签中添加。

**Service** 要由 **Activity** 通过 **startService** 或者 **bindService** 来启动, **Intent** 负责传递参数。先贴出本文程序运行截图:



本文主要讲解 **Service** 的调用，以及其生命周期。

Log		
.d	tag	Message
I9	Service	onCreate
I9	Service	onStart
I9	Service	onDestroy

上图是 **startService** 之后再 **stopService** 的 **Service** 状态变化。

Log		
.d	tag	Message
I9	Service	onCreate
I9	Service	onBind
I9	Service	onUnbind
I9	Service	onDestroy

上图是 **bindService** 之后再 **unbindService** 的 **Service** 状态变化。

`startService` 与 `bindService` 都可以启动 `Service`，那么它们之间有什么区别呢？它们两者的区别就是使 `Service` 的周期改变。由 `startService` 启动的 `Service` 必须要有 `stopService` 来结束 `Service`，不调用 `stopService` 则会造成 `Activity` 结束了而 `Service` 还运行着。`bindService` 启动的 `Service` 可以由 `unbindService` 来结束，也可以在 `Activity` 结束之后(`onDestroy`)自动结束。

Log		
id	tag	Message
59	Service	onCreate
59	Service	onStart
59	Activity	onDestroy

上图是 `startService` 之后再 `Activity.finish()` 的 `Service` 状态变化，`Service` 还在跑着。

tag	Message
Service	onCreate
Service	onBind
Activity	onDestroy
ActivityThread	Activity com.testSe:
ActivityThread	android.app.Service
ActivityThread	at android.app.
ActivityThread	at android.app.
ActivityThread	at android.app.
ActivityThread	at android.cont
ActivityThread	at com.testServ:
ActivityThread	at android.view
ActivityThread	at android.view
ActivityThread	at android.os.H
ActivityThread	at android.os.H
ActivityThread	at android.os.L
ActivityThread	at android.app.
ActivityThread	at java.lang.re:
ActivityThread	at java.lang.re:
ActivityThread	at com.android.:
ActivityThread	at com.android.:
ActivityThread	at dalvik.syste
Service	onUnbind
Service	onDestroy

上图是 `bindService` 之后再 `Activity.finish()` 的 `Service` 状态变化，  
`Service` 最后自动 `unbindService`。

main.xml 代码:

[view plaincopy to clipboardprint?](#)

```

1.  <?xml version="1.0" encoding="utf-8"?>
2.  <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
3.      android:orientation="vertical" android:layout_width="fill_parent"
4.      android:layout_height="fill_parent">
5.      <Button android:layout_width="wrap_content"
6.          android:layout_height="wrap_content" android:id="@+id/btnStartMyServ
7.          ice"
8.          android:text="StartMyService"></Button>
9.      <Button android:layout_width="wrap_content"
10.         android:layout_height="wrap_content" android:id="@+id/btnStopMyServi
11.         ce"
12.         android:text="StopMyService"></Button>
13.      <Button android:layout_width="wrap_content"

```

```

12.         android:layout_height="wrap_content" android:id="@+id/btnBindMyService"
13.         android:text="BindMyService"></Button>
14.     <Button android:layout_width="wrap_content"
15.         android:layout_height="wrap_content" android:id="@+id/btnUnbindMyService"
16.         android:text="UnbindMyService"></Button>
17.     <Button android:layout_width="wrap_content"
18.         android:layout_height="wrap_content" android:id="@+id/btnExit"
19.         android:text="退出程序"></Button>
20. </LinearLayout>

```

testService.java 的源码:

[view plaincopy to clipboardprint?](#)

```

1. package com.testService;
2.
3. import android.app.Activity;
4. import android.app.Service;
5. import android.content.ComponentName;
6. import android.content.Intent;
7. import android.content.ServiceConnection;
8. import android.os.Bundle;
9. import android.os.IBinder;
10. import android.util.Log;
11. import android.view.View;
12. import android.widget.Button;
13.
14. public class testService extends Activity {
15.     Button btnStartMyService,btnStopMyService,btnBindMyService,btnUnbindMyService,btnExit;
16.     @Override
17.     public void onCreate(Bundle savedInstanceState) {
18.         super.onCreate(savedInstanceState);
19.         setContentView(R.layout.main);
20.         btnStartMyService=(Button)this.findViewById(R.id.btnStartMyService);
21.
22.         btnStartMyService.setOnClickListener(new ClickEvent());
23.
24.         btnStopMyService=(Button)this.findViewById(R.id.btnStopMyService);
25.         btnStopMyService.setOnClickListener(new ClickEvent());
26.
27.         btnBindMyService=(Button)this.findViewById(R.id.btnBindMyService);

```

```

27.         btnBindMyService.setOnClickListener(new ClickEvent());
28.
29.         btnUnbindMyService=(Button)this.findViewById(R.id.btnUnbindMyService
    );
30.         btnUnbindMyService.setOnClickListener(new ClickEvent());
31.
32.         btnExit=(Button)this.findViewById(R.id.btnExit);
33.         btnExit.setOnClickListener(new ClickEvent());
34.     }
35.     @Override
36.     public void onDestroy()
37.     {
38.         super.onDestroy();
39.         Log.e("Activity", "onDestroy");
40.     }
41.
42.     private ServiceConnection _connection = new ServiceConnection() {
43.         @Override
44.         public void onServiceConnected(ComponentName arg0, IBinder arg1) {
45.             // TODO Auto-generated method stub
46.         }
47.
48.         @Override
49.         public void onServiceDisconnected(ComponentName name) {
50.             // TODO Auto-generated method stub
51.         }
52.     };
53.     class ClickEvent implements View.OnClickListener{
54.
55.         @Override
56.         public void onClick(View v) {
57.             Intent intent=new Intent(testService.this,MyService.class);
58.             if(v==btnStartMyService){
59.                 testService.this.startService(intent);
60.             }
61.             else if(v==btnStopMyService){
62.                 testService.this.stopService(intent);
63.             }
64.             else if(v==btnBindMyService){
65.                 testService.this.bindService(intent, _connection, Service.BI
ND_AUTO_CREATE);
66.             }
67.             else if(v==btnUnbindMyService){

```

```

68.         if(MyService.ServiceState=="onBind");//Service 绑定了之后才能解
        绑
69.         testService.this.unbindService(_connection);
70.     }
71.     else if(v==btnExit)
72.     {
73.         testService.this.finish();
74.     }
75.
76.     }
77.
78. }
79. }

```

## MyService.java 的源码:

[view plaincopy to clipboardprint?](#)

```

1. package com.testService;
2.
3. import android.app.Service;
4. import android.content.Intent;
5. import android.os.IBinder;
6. import android.util.Log;
7.
8. public class MyService extends Service {
9.     static public String ServiceState="";
10.    @Override
11.    public IBinder onBind(Intent arg0) {
12.        Log.e("Service", "onBind");
13.        ServiceState="onBind";
14.        return null;
15.    }
16.    @Override
17.    public boolean onUnbind(Intent intent){
18.        super.onUnbind(intent);
19.        Log.e("Service", "onUnbind");
20.        ServiceState="onUnbind";
21.        return false;
22.    }
23.    }
24.    @Override
25.    public void onCreate(){
26.        super.onCreate();

```

```
27.         Log.e("Service", "onCreate");
28.         ServiceState="onCreate";
29.     }
30.     @Override
31.     public void onDestroy(){
32.         super.onDestroy();
33.         Log.e("Service", "onDestroy");
34.         ServiceState="onDestroy";
35.     }
36.     @Override
37.     public void onStart(Intent intent,int startid){
38.         super.onStart(intent, startid);
39.         Log.e("Service", "onStart");
40.         ServiceState="onStart";
41.     }
42.
43. }
```





本文档来自 **安卓巴士** ([www.apkbus.com](http://www.apkbus.com)) 整理总结

## 作者简介:

---

张国威 (用户名: hellogv)

<http://hi.csdn.net/hellogv>

- 性别: 男
- 生日: 1985 年 11 月 20 日
- 婚恋: 单身
- 居住: 江苏 南京
- 家乡: 广东 湛江
- MSN: hellogv@qq.com
- Email: hellogv@QQ.com

## Android提高第六篇之BroadcastReceiver

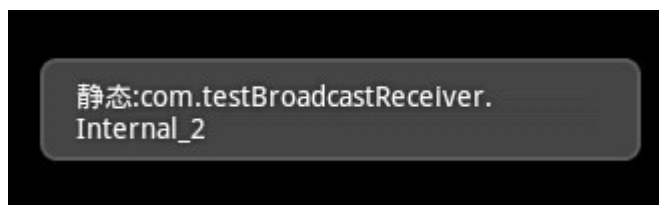
---

前面分别讨论了 [Activity](#) 和 [Service](#), 这次就轮到 **BroadcastReceiver**, **Broadcast** 是应用程序间通信的手段。**BroadcastReceiver** 也是跟 **Intent** 紧密相连的, 动态/静态注册了 **BroadcastReceiver** 之后, 使用 **sendBroadcast** 把 **Intent** 发送之后, 系统会自动把符合条件的 **BroadcastReceiver** 启动, 跟嵌入式系统的中断类似。

本文主要演示了如何静态/动态注册 **BroadcastReceiver**, 向系统索取电量信息, 以及枚举信息的字段。本文运行截图如下:



上图是发送 Intent 至内部动态注册的 BroadcastReceiver，接收到之后显示消息名称。  
动态注册 BroadcastReceiver 用到 registerReceiver()。



上图是发送 Intent 至内部静态注册的 BroadcastReceiver，接收到之后显示消息名称。  
静态注册比动态注册麻烦点，先新建一个类继承 BroadcastReceiver，然后到 AndroidManifest.xml 添加

[view plaincopy to clipboardprint?](#)

```

1. <receiver android:name="clsReceiver2">
2.     <intent-filter>
3.         <action
4.             android:name="com.testBroadcastReceiver.Internal_2"/>
5.     </intent-filter>
6. </receiver>

```

第一个 name 是类名，第二个是 action 的名称。

Log	
tag	Message
icon-small	17302169
scale	100
present	true
technology	Li-ion
level	50
voltage	0
status	2
plugged	1
health	2
temperature	0
Name	hellogv
Blog	http://blog.csdn.net/hellogv

上图是枚举 **Intent** 消息的字段，这个功能比较适合懒人，把收到的 **Intent** 消息的字段全部分解了，再看看哪个需要的，懒得记住。实现这部分的代码如下：

[view plaincopy to clipboardprint?](#)

```

1. //当未知 Intent 包含的内容，则需要通过以下方法来列举
2.         Bundle b=intent.getExtras();
3.         Object[] lstName=b.keySet().toArray();
4.
5.         for(int i=0;i<lstName.length;i++)
6.         {
7.             String keyName=lstName[i].toString();
8.             Log.e(keyName,String.valueOf(b.get(keyName)));
9.         }

```

main.xml 的代码如下：

[view plaincopy to clipboardprint?](#)

```

1. <?xml version="1.0" encoding="utf-8"?>
2. <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
3.     android:orientation="vertical" android:layout_width="fill_parent"
4.     android:layout_height="fill_parent">
5.
6.     <Button android:id="@+id/Button01" android:layout_width="wrap_content"

```

```

7.         android:layout_height="wrap_content" android:text="发送至内部动态注册
    的 BroadcastReceiver"></Button>
8.     <Button android:id="@+id/Button02" android:layout_width="wrap_content"
9.         android:layout_height="wrap_content" android:text="发送至内部静态注册
    BroadcastReceiver"></Button>
10.    <Button android:id="@+id/Button03" android:layout_width="wrap_content"
11.        android:layout_height="wrap_content" android:text="发送至系统
    BroadcastReceiver"></Button>
12. </LinearLayout>

```

testBroadcastReceiver.java 的代码如下:

[view plaincopy to clipboardprint?](#)

```

1. package com.testBroadcastReceiver;
2.
3. import android.app.Activity;
4. import android.content.BroadcastReceiver;
5. import android.content.Context;
6. import android.content.Intent;
7. import android.content.IntentFilter;
8. import android.os.Bundle;
9. import android.util.Log;
10. import android.view.View;
11. import android.widget.Button;
12. import android.widget.Toast;
13.
14. public class testBroadcastReceiver extends Activity {
15.     Button btnInternal1,btnInternal2,btnSystem;
16.     static final String INTENAL_ACTION_1 = "com.testBroadcastReceiver.Intern
    al_1";
17.     static final String INTENAL_ACTION_2 = "com.testBroadcastReceiver.Intern
    al_2";
18.     static final String INTENAL_ACTION_3 = "com.testBroadcastReceiver.Intern
    al_3";
19.     @Override
20.     public void onCreate(Bundle savedInstanceState) {
21.         super.onCreate(savedInstanceState);
22.         setContentView(R.layout.main);
23.         btnInternal1=(Button)this.findViewById(R.id.Button01);
24.         btnInternal1.setOnClickListener(new ClickEvent());
25.         btnInternal2=(Button)this.findViewById(R.id.Button02);
26.         btnInternal2.setOnClickListener(new ClickEvent());
27.         btnSystem=(Button)this.findViewById(R.id.Button03);

```

```

28.         btnSystem.setOnClickListener(new ClickEvent());
29.         //动态注册广播消息
30.         registerReceiver(bcrIntenal1, new IntentFilter(INTENAL_ACTION_1));
31.     }
32.     class ClickEvent implements View.OnClickListener{
33.
34.         @Override
35.         public void onClick(View v) {
36.             if(v==btnInternal1)//给动态注册的 BroadcastReceiver 发送数据
37.             {
38.                 Intent intent = new Intent(INTENAL_ACTION_1);
39.                 sendBroadcast(intent);
40.             }
41.             else if(v==btnInternal2)//给静态注册的 BroadcastReceiver 发送数据
42.             {
43.                 Intent intent = new Intent(INTENAL_ACTION_2);
44.                 sendBroadcast(intent);
45.             }
46.             else if(v==btnSystem)//动态注册 接收 2 组信息的 BroadcastReceiver
47.             {
48.                 IntentFilter filter = new IntentFilter();//
49.                 filter.addAction(Intent.ACTION_BATTERY_CHANGED);//系统电量检
测信息
50.                 filter.addAction(INTENAL_ACTION_3);//第三组自定义消息
51.                 registerReceiver(batInfoReceiver, filter);
52.
53.                 Intent intent = new Intent(INTENAL_ACTION_3);
54.                 intent.putExtra("Name", "hellogv");
55.                 intent.putExtra("Blog", "http://blog.csdn.net/hellogv");
56.                 sendBroadcast(intent);//传递过去
57.             }
58.         }
59.
60.     }
61.
62.     /*
63.      * 接收动态注册广播的 BroadcastReceiver
64.      */
65.     private BroadcastReceiver bcrIntenal1 = new BroadcastReceiver() {
66.
67.         public void onReceive(Context context, Intent intent) {
68.             String action = intent.getAction();
69.             Toast.makeText(context, "动态:"+action, 1000).show();
70.         }

```

```

71.     };
72.
73.
74.     private BroadcastReceiver batInfoReceiver = new BroadcastReceiver() {
75.
76.         public void onReceive(Context context, Intent intent) {
77.             String action = intent.getAction();
78.             //如果捕捉到的 action 是 ACTION_BATTERY_CHANGED
79.             if (Intent.ACTION_BATTERY_CHANGED.equals(action)) {
80.                 //当未知 Intent 包含的内容，则需要通过以下方法来列举
81.                 Bundle b=intent.getExtras();
82.                 Object[] lstName=b.keySet().toArray();
83.
84.                 for(int i=0;i<lstName.length;i++)
85.                 {
86.                     String keyName=lstName[i].toString();
87.                     Log.e(keyName,String.valueOf(b.get(keyName)));
88.                 }
89.             }
90.             //如果捕捉到的 action 是 INTENAL_ACTION_3
91.             if (INTENAL_ACTION_3.equals(action)) {
92.                 //当未知 Intent 包含的内容，则需要通过以下方法来列举
93.                 Bundle b=intent.getExtras();
94.                 Object[] lstName=b.keySet().toArray();
95.
96.                 for(int i=0;i<lstName.length;i++)
97.                 {
98.                     String keyName=lstName[i].toString();
99.                     Log.e(keyName,b.getString(keyName));
100.                }
101.            }
102.        }
103.    };
104.
105.
106. }

```

clsReceiver2.java 的代码如下：

[view plaincopy to clipboardprint?](#)

```

1. package com.testBroadcastReceiver;
2.
3. import android.content.BroadcastReceiver;

```

```

4. import android.content.Context;
5. import android.content.Intent;
6. import android.widget.Toast;
7.
8. /*
9.  * 接收静态注册广播的 BroadcastReceiver,
10.  * step1:要到 AndroidManifest.xml 这里注册消息
11.  *      <receiver android:name="clsReceiver2">
12.  *          <intent-filter>
13.  *              <action
14.  *                  android:name="com.testBroadcastReceiver.Internal_2"/>
15.  *          </intent-filter>
16.  *      </receiver>
17.  * step2:定义消息的字符串
18.  * step3:通过 Intent 传递消息来驱使 BroadcastReceiver 触发
19.  */
20. public class clsReceiver2 extends BroadcastReceiver{
21.     @Override
22.     public void onReceive(Context context, Intent intent) {
23.         String action = intent.getAction();
24.         Toast.makeText(context, "静态:"+action, 1000).show();
25.
26.     }
27. }

```



本文档来自 **安卓巴士** ([www.apkbus.com](http://www.apkbus.com)) 整理总结

## 作者简介:

张国威 (用户名: hellogv)

<http://hi.csdn.net/hellogv>

- 性别: 男
- 生日: 1985 年 11 月 20 日
- 婚恋: 单身
- 居住: 江苏 南京
- 家乡: 广东 湛江
- MSN: hellogv@qq.com
- Email: hellogv@QQ.com

## Android提高第七篇之XML解析与生成

本文使用 **SAX** 来解析 **XML**, 在 **Android** 里面可以使用 **SAX** 和 **DOM**, **DOM** 需要把整个 **XML** 文件读入内存再解析, 比较消耗内存, 而 **SAX** 基于事件驱动的处理方式, 可以在各节点触发回调函数, 不过 **SAX** 适合节点结构简单的 **XML** 文档, 复杂的 **XML** 文档在后期的节点深度处理会有点麻烦。

本文要解析的 **test.xml** 文件如下:

[view plaincopy to clipboardprint?](#)

```
1. <?xml version="1.0" encoding="utf-8"?>
2. <test>
3.   <title>testSAX</title>
4.   <content aa="1" bb="2">
5.     <name>hellogv</name>
6.     <url>http://blog.csdn.net/hellogv</url>
```



```
7.     </content>
8. </test>
```

解析如上 XML 的结果如下：

Log	
tag	Message
startTag	test
startTag	title
text	testSAX
endTag	title
startTag	content
Attr	aa=1
Attr	bb=2
startTag	name
text	hellogv
endTag	name
startTag	url
text	http://blog.csdn.net/hellogv
endTag	url
endTag	content
endTag	test

使用 SAX 解析，需要定义 **SAXParserFactory**(使应用程序能够配置和获取基于 SAX 的解析器以解析 XML 文档), **SAXParser**(从各种输入源解析 XML), **XMLReader**(使用回调函数读取 XML 文档)，其中 **XMLReader** 是个关键。**XMLReader** 可以为解析 XML 定义各种回调函数，“条件符合”的时候触发这些回调函数。

[view plain](#)[copy](#) to clipboardprint?

```
1. SAXParserFactory factory = SAXParserFactory.newInstance();
2. SAXParser parser = factory.newSAXParser();
3. XMLReader reader = parser.getXMLReader();
4. reader.setContentHandler(handler);
5. reader.parse(new InputSource(testSAX.this.getResources()
6.     .openRawResource(R.raw.test)));
```

在这段代码里，XMLReader 就调用继承 DefaultHandler 的 SAXHandler。DefaultHandler 已实现 ContentHandler, DTDHandler, EntityResolver, ErrorHandler 等接口，包含常见读取 XML 的操作，具体请看下面的 SAXHandler.java 源码。

生成 XML 的结果如下：



上图是读取各节点之后，使用 XmlSerializer 重新组合并输出 XML 字符串。

本文的 main.xml 代码如下：

[view plaincopy to clipboardprint?](#)

```

1. <?xml version="1.0" encoding="utf-8"?>
2. <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
3.     android:orientation="vertical" android:layout_width="fill_parent"
4.     android:layout_height="fill_parent">
5.
6.     <Button android:layout_height="wrap_content"
7.         android:layout_width="fill_parent" android:id="@+id/btnSAX"
8.         android:text="使用 SAX 解析 XML"></Button>
9.     <Button android:layout_height="wrap_content"
10.        android:layout_width="fill_parent" android:text="生成
XML" android:id="@+id/btnOutput"></Button>
11.    <EditText android:text="@+id/EditText01" android:id="@+id/EditText01"
12.        android:layout_width="fill_parent" android:layout_height="fill_paren
t"></EditText>
13.
14. </LinearLayout>

```

SAXHandler.java 的源码如下：

[view plaincopy to clipboardprint?](#)

```

1. package com.testSAX;
2.
3. import java.util.ArrayList;
4. import org.xml.sax.Attributes;
5. import org.xml.sax.SAXException;
6. import org.xml.sax.helpers.DefaultHandler;
7.
8. import android.util.Log;
9.
10. public class SAXHandler extends DefaultHandler{
11.     private ArrayList<String> keys = new ArrayList<String>();//保存字段名
    称
12.     private ArrayList<Object> values = new ArrayList<Object>();//保存值
13.     @Override
14.     public void startDocument() throws SAXException {
15.         super.startDocument();
16.
17.     }
18.
19.     @Override
20.     public void endDocument() throws SAXException {
21.         super.endDocument();
22.     }

```

```

23.
24.     @Override
25.     public void startElement(String uri, String localName, String qName,
26.         Attributes attributes) throws SAXException {
27.         //保存开始标记
28.         keys.add("startTag");
29.         values.add(localName);
30.
31.         Log.e("startTag",localName);
32.         //保存属性值
33.         for ( int i = 0; i < attributes.getLength(); i++ ){
34.             keys.add("Attr");
35.             String[] str=new String[2];
36.             str[0]=attributes.getLocalName(i);
37.             str[1]=attributes.getValue(i);
38.             values.add(str);
39.             Log.e("Attr",str[0]+"="+str[1]);
40.         }
41.     }
42.
43.     @Override
44.     public void endElement(String uri, String localName, String qName)
45.         throws SAXException {
46.         //保存结束标记
47.         keys.add("endTag");
48.         values.add(localName);
49.         Log.e("endTag",localName);
50.     }
51.
52.     @Override
53.     public void characters(char[] ch, int start, int length)
54.         throws SAXException {
55.         String value = new String(ch, start, length);
56.         value = value.trim();
57.         if (value.length() == 0)
58.             return;
59.
60.         keys.add("text");
61.         values.add(value);
62.         Log.e("text",value);
63.     }
64.
65.     public ArrayList<String> GetKeys()

```

```

66.     {
67.         return keys;
68.     }
69.
70.     public ArrayList<Object> GetValues()
71.     {
72.         return values;
73.     }
74.
75.
76. }

```

testSAX.java 的源码如下:

[view plain](#)[copy to clipboard](#)[print?](#)

```

1.  package com.testSAX;
2.
3.  import java.io.StringWriter;
4.
5.  import javax.xml.parsers.SAXParser;
6.  import javax.xml.parsers.SAXParserFactory;
7.
8.  import org.xml.sax.InputSource;
9.  import org.xml.sax.XMLReader;
10. import org.xmlpull.v1.XmlSerializer;
11.
12. import android.app.Activity;
13. import android.os.Bundle;
14.
15. import android.util.Xml;
16. import android.view.View;
17. import android.widget.Button;
18. import android.widget.EditText;
19.
20. public class testSAX extends Activity {
21.     /** Called when the activity is first created. */
22.     Button btnSAX, btnOutput;
23.     EditText memo;
24.     SAXHandler handler = new SAXHandler();
25.
26.
27.     @Override
28.     public void onCreate(Bundle savedInstanceState) {

```

```

29.         super.onCreate(savedInstanceState);
30.         setContentView(R.layout.main);
31.         btnSAX = (Button) this.findViewById(R.id.btnSAX);
32.         btnSAX.setOnClickListener(new ClickEvent());
33.         btnOutput = (Button) this.findViewById(R.id.btnOutput);
34.         btnOutput.setOnClickListener(new ClickEvent());
35.         memo = (EditText) this.findViewById(R.id.EditText01);
36.
37.     }
38.
39.     class ClickEvent implements View.OnClickListener {
40.
41.         @Override
42.         public void onClick(View v) {
43.             if (v == btnSAX) { //解析 XML, 并保存标记, 属性等值
44.                 try {
45.                     SAXParserFactory factory = SAXParserFactory.newInstance(
46. );
47.                     SAXParser parser = factory.newSAXParser();
48.                     XMLReader reader = parser.getXMLReader();
49.                     reader.setContentHandler(handler);
50.                     reader.parse(new InputSource(testSAX.this.getResources()
51. ).openRawResource(R.raw.test)));
52.                 } catch (Exception ee) {}
53.             }
54.             else if (v == btnOutput) { //生成 XML
55.                 try {
56.                     XmlSerializer serializer = Xml.newSerializer();
57.                     StringWriter writer = new StringWriter();
58.                     try {
59.                         serializer.setOutput(writer);
60.                         serializer.startDocument("UTF-8", true);
61.
62.                         for(int i=0; i<handler.GetKeys().size(); i++)
63.                         {
64.                             if(handler.GetKeys().get(i).equals("startTag"))
65.                             {
66.                                 serializer.startTag("", (String) handler.Get
67. Values().get(i));
68.                             }
69.                             else if(handler.GetKeys().get(i).equals("Attr"))
70.                             {

```

```

68.                String[] str= (String[]) handler.GetValues()
        .get(i);
69.                serializer.attribute("",str[0],str[1]);
70.            }
71.            else if(handler.GetKeys().get(i).equals("text"))
72.                serializer.text((String)handler.GetValues().
        get(i));
73.            else if(handler.GetKeys().get(i).equals("endTag"
        ))
74.            {
75.                serializer.endTag("", (String) handler.GetVa
        lues().get(i));
76.            }
77.        }
78.        serializer.endDocument();
79.        String text=writer.toString();
80.        text=text.replace("><", ">\r\n<");
81.        memo.setText(text);//输出到文本框
82.    } catch (Exception e) {
83.        throw new RuntimeException(e);
84.    }
85.
86.    } catch (Exception e) {}
87.    }
88.
89.    }
90.
91.    }
92. }

```



本文档来自 **安卓巴士** ([www.apkbus.com](http://www.apkbus.com)) 整理总结

## 作者简介:

---

张国威 (用户名: hellogv)

<http://hi.csdn.net/hellogv>

- 性别: 男
- 生日: 1985 年 11 月 20 日
- 婚恋: 单身
- 居住: 江苏 南京
- 家乡: 广东 湛江
- MSN: hellogv@qq.com
- Email: hellogv@QQ.com

## Android提高第八篇之SQLite分页读取

---

**Android** 包含了常用于嵌入式系统的 **SQLite**, 免去了开发者自己移植安装功夫。**SQLite** 支持多数 **SQL92** 标准, 很多常用的 **SQL** 命令都能在 **SQLite** 上面使用, 除此之外 **Android** 还提供了一系列自定义的方法去简化对 **SQLite** 数据库的操作。不过有跨平台需求的程序就建议使用标准的 **SQL** 语句, 毕竟这样容易在多个平台之间移植。

先贴出本文程序运行的结果:





本文主要讲解了SQLite的基本用法，如：创建数据库，使用SQL命令查询数据表、插入数据，关闭数据库，以及使用GridView实现了一个分页栏(关于GridView的用法)，用于把数据分页显示。

分页栏的 pagebuttons.xml 的源码如下：

[view plain](#) [copy](#) to clipboardprint?

```
1. <?xml version="1.0" encoding="utf-8"?>
2. <RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
3.     android:layout_height="wrap_content" android:paddingBottom="4dip"
4.     android:layout_width="fill_parent">
5.     <TextView android:layout_width="wrap_content"
6.         android:layout_below="@+id/ItemImage" android:layout_height="wrap_co
7.         ntent"
8.         android:text="TextView01" android:layout_centerHorizontal="true"
9.         android:id="@+id/ItemText">
```

```
9.         </TextView>
10. </RelativeLayout>
```

main.xml 的源码如下:

[view plaincopy to clipboardprint?](#)

```
1. <?xml version="1.0" encoding="utf-8"?>
2. <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
3.     android:orientation="vertical" android:layout_width="fill_parent"
4.     android:layout_height="fill_parent">
5.     <Button android:layout_height="wrap_content"
6.         android:layout_width="fill_parent" android:id="@+id/btnCreateDB"
7.         android:text="创建数据库"></Button>
8.     <Button android:layout_height="wrap_content"
9.         android:layout_width="fill_parent" android:text="插入一串实验数据
    " android:id="@+id/btnInsertRec"></Button>
10.    <Button android:layout_height="wrap_content" android:id="@+id/btnClose"
11.        android:text="关闭数据库
    " android:layout_width="fill_parent"></Button>
12.    <EditText android:text="@+id/EditText01" android:id="@+id/EditText01"
13.        android:layout_width="fill_parent" android:layout_height="256dip"></
    EditText>
14.    <GridView android:id="@+id/gridview" android:layout_width="fill_parent"
15.        android:layout_height="32dip" android:numColumns="auto_fit"
16.        android:columnWidth="40dip"></GridView>
17. </LinearLayout>
```

本文程序源码如下:

[view plaincopy to clipboardprint?](#)

```
1. package com.testSQLite;
2.
3. import java.util.ArrayList;
4. import java.util.HashMap;
5. import android.app.Activity;
6. import android.database.Cursor;
7. import android.database.SQLException;
8. import android.database.sqlite.SQLiteDatabase;
```

```

9. import android.os.Bundle;
10. import android.util.Log;
11. import android.view.View;
12. import android.widget.AdapterView;
13. import android.widget.AdapterView.OnItemClickListener;
14. import android.widget.Button;
15. import android.widget.EditText;
16. import android.widget.GridView;
17. import android.widget.SimpleAdapter;
18.
19. public class testSQLite extends Activity {
20.     /** Called when the activity is first created. */
21.     Button btnCreateDB, btnInsert, btnClose;
22.     EditText edtSQL;//显示分页数据
23.     SQLiteDatabase db;
24.     int id;//添加记录时的 id 累加标记, 必须全局
25.     static final int PageSize=10;//分页时, 每页的数据总数
26.     private static final String TABLE_NAME = "stu";
27.     private static final String ID = "id";
28.     private static final String NAME = "name";
29.
30.     SimpleAdapter saPageID;// 分页栏适配器
31.     ArrayList<HashMap<String, String>> lstPageID;// 分页栏的数据源, 与 PageSize
    和数据总数相关
32.
33.     @Override
34.     public void onCreate(Bundle savedInstanceState) {
35.         super.onCreate(savedInstanceState);
36.         setContentView(R.layout.main);
37.         btnCreateDB = (Button) this.findViewById(R.id.btnCreateDB);
38.         btnCreateDB.setOnClickListener(new ClickEvent());
39.
40.         btnInsert = (Button) this.findViewById(R.id.btnInsertRec);
41.         btnInsert.setOnClickListener(new ClickEvent());
42.
43.         btnClose = (Button) this.findViewById(R.id.btnClose);
44.         btnClose.setOnClickListener(new ClickEvent());
45.
46.         edtSQL=(EditText)this.findViewById(R.id.EditText01);
47.
48.         GridView gridview = (GridView) findViewById(R.id.gridview);//分页栏控
    件
49.         // 生成动态数组, 并且转入数据
50.         lstPageID = new ArrayList<HashMap<String, String>>();

```

```

51.
52.     // 生成适配器的 ImageItem <====> 动态数组的元素，两者一一对应
53.     saPageID = new SimpleAdapter(testSQLite.this, // 没什么解释
54.         lstPageID, // 数据来源
55.         R.layout.pagebuttons, // XML 实现
56.         new String[] { "ItemText" },
57.         new int[] { R.id.ItemText });
58.
59.     // 添加并且显示
60.     gridView.setAdapter(saPageID);
61.     // 添加消息处理
62.     gridView.setOnItemClickListener(new OnItemClickListener(){
63.
64.         @Override
65.         public void onItemClick(AdapterView<?> arg0, View arg1, int arg2
66.             ,
67.             long arg3) {
68.             LoadPage(arg2); // 根据所选分页读取对应的数据
69.         }
70.     });
71. }
72.
73.
74. class ClickEvent implements View.OnClickListener {
75.
76.     @Override
77.     public void onClick(View v) {
78.         if (v == btnCreateDB) {
79.             CreateDB();
80.         } else if (v == btnInsert) {
81.             InsertRecord(16); // 插入 16 条记录
82.             RefreshPage();
83.         } else if (v == btnClose) {
84.             db.close();
85.         }
86.     }
87.
88. }
89.
90.
91. /*
92.  * 读取指定 ID 的分页数据
93.  * SQL:Select * From TABLE_NAME Limit 9 Offset 10;

```

```

94.      * 表示从 TABLE_NAME 表获取数据, 跳过 10 行, 取 9 行
95.      */
96.      void LoadPage(int pageID)
97.      {
98.          String sql= "select * from " + TABLE_NAME +
99.              " Limit "+String.valueOf(PageSize)+ " Offset " +String.valueOf(pageI
              D*PageSize);
100.          Cursor rec = db.rawQuery(sql, null);
101.
102.          setTitle("当前分页的数据总数:"+String.valueOf(rec.getCount()));
103.
104.          // 取得字段名称
105.          String title = "";
106.          int colCount = rec.getColumnCount();
107.          for (int i = 0; i < colCount; i++)
108.              title = title + rec洗getColumnName(i) + "      ";
109.
110.
111.          // 列举出所有数据
112.          String content="";
113.          int recCount=rec.getCount();
114.          for (int i = 0; i < recCount; i++) {//定位到一条数据
115.              rec.moveToPosition(i);
116.              for(int ii=0;ii<colCount;ii++){//定位到一条数据中的每个字段
117.                  {
118.                      content=content+rec.getString(ii)+"      ";
119.                  }
120.                      content=content+"\r\n";
121.              }
122.
123.              edtSQL.setText(title+"\r\n"+content);//显示出来
124.              rec.close();
125.          }
126.
127.      /*
128.      * 在内存创建数据库和数据表
129.      */
130.      void CreateDB() {
131.          // 在内存创建数据库
132.          db = SQLiteDatabase.create(null);
133.          Log.e("DB Path", db.getPath());
134.          String amount = String.valueOf(databaseList().length);
135.          Log.e("DB amount", amount);
136.          // 创建数据表

```

```

137.         String sql = "CREATE TABLE " + TABLE_NAME + " (" + ID
138.             + " text not null, " + NAME + " text not null " + ");";
139.         try {
140.             db.execSQL("DROP TABLE IF EXISTS " + TABLE_NAME);
141.             db.execSQL(sql);
142.         } catch (SQLException e) {}
143.     }
144.
145.     /*
146.     * 插入N条数据
147.     */
148.     void InsertRecord(int n) {
149.         int total = id + n;
150.         for (; id < total; id++) {
151.             String sql = "insert into " + TABLE_NAME + " (" + ID + ", " + N
AME
152.                 + ") values('" + String.valueOf(id) + "', 'test');"
153.             try {
154.                 db.execSQL(sql);
155.             } catch (SQLException e) {}
156.         }
157.     }
158. }
159.
160. /*
161. * 插入之后刷新分页
162. */
163. void RefreshPage()
164. {
165.     String sql = "select count(*) from " + TABLE_NAME;
166.     Cursor rec = db.rawQuery(sql, null);
167.     rec.moveToLast();
168.     long recSize=rec.getLong(0);//取得总数
169.     rec.close();
170.     int pageNum=(int)(recSize/PageSize) + 1;//取得分页数
171.
172.     lstPageID.clear();
173.     for (int i = 0; i < pageNum; i++) {
174.         HashMap<String, String> map = new HashMap<String, String>();
175.
176.         map.put("ItemText", "No." + String.valueOf(i));
177.
178.         lstPageID.add(map);

```

```
178.     }  
179.     saPageID.notifyDataSetChanged();  
180. }  
181. }
```



本文档来自 **安卓巴士** ([www.apkbus.com](http://www.apkbus.com)) 整理总结

## 作者简介:

---

张国威 (用户名: hellogv)

<http://hi.csdn.net/hellogv>

- 性别: 男
- 生日: 1985 年 11 月 20 日
- 婚恋: 单身
- 居住: 江苏 南京
- 家乡: 广东 湛江
- MSN: hellogv@qq.com
- Email: hellogv@QQ.com

## Android提高第九篇之SQLite分页表格

---

上次讲的**Android**上的**SQLite**分页读取, 只用文本框显示数据而已, 这次就讲得更加深入些, 实现并封装一个**SQL**分页表格控件, 不仅支持分页还是以表格的形式展示数据。先来看看本文程序运行的动画:





这个 **SQL** 分页表格控件主要分为“表格区”和“分页栏”这两部分，这两部分都是基于 **GridView** 实现的。网上介绍 **Android** 上实现表格的 **DEMO** 一般都用 **ListView**。**ListView** 与 **GridView** 对比，**ListView** 最大的优势是格单元的大小可以自定义，可以某单元长某单元短，但是难于实现自适应数据表的结构；而 **GridView** 最大的优势就是自适应数据表的结构，但是格单元统一大小。。。对于数据表结构多变的情况，建议使用 **GridView** 实现表格。

本文实现的 **SQL** 分页表格控件有以下特点：

- 1.自适应数据表结构，但是格单元统一大小；

2.支持分页;

3."表格区"有按键事件回调处理, "分页栏"有分页切换事件回调处理。

本文程序代码较多, 可以到这里下载整个工程的源

码: <http://www.rayfile.com/files/72e78b68-f2e5-11df-8469-0015c55db73d/>

items.xml 的代码如下, 它是"表格区"和"分页栏"的格单元实现:

[view plaincopy to clipboardprint?](#)

```
1. <?xml version="1.0" encoding="utf-8"?>
2. <LinearLayout android:id="@+id/LinearLayout01"
3.     xmlns:android="http://schemas.android.com/apk/res/android"
4.     android:layout_width="fill_parent" android:background="#555555"
5.     android:layout_height="wrap_content">
6.     <TextView android:layout_below="@+id/ItemImage" android:text="TextView01"
7.         android:id="@+id/ItemText" android:bufferType="normal"
8.         android:singleLine="true" android:background="#000000"
9.         android:layout_width="fill_parent" android:gravity="center"
10.        android:layout_margin="1dip" android:layout_gravity="center"
11.        android:layout_height="wrap_content">
12.     </TextView>
13. </LinearLayout>
```

main.xml 的代码如下:

[view plaincopy to clipboardprint?](#)

```
1. <?xml version="1.0" encoding="utf-8"?>
2. <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
3.     android:orientation="vertical" android:layout_width="fill_parent"
4.     android:layout_height="fill_parent" android:id="@+id/MainLinearLayout">
5.     <Button android:layout_height="wrap_content"
6.         android:layout_width="fill_parent" android:id="@+id/btnCreateDB"
7.         android:text="创建数据库"></Button>
8.     <Button android:layout_height="wrap_content"
```

```

9.         android:layout_width="fill_parent" android:text="插入一串实验数据
    " android:id="@+id/btnInsertRec"></Button>
10.     <Button android:layout_height="wrap_content" android:id="@+id/btnClose"

11.         android:text="关闭数据库
    " android:layout_width="fill_parent"></Button>
12. </LinearLayout>

```

演示程序 testSQLite.java 的源码:

[view plaincopy to clipboardprint?](#)

```

1. package com.testSQLite;
2. import android.app.Activity;
3. import android.database.Cursor;
4. import android.database.SQLException;
5. import android.database.sqlite.SQLiteDatabase;
6. import android.os.Bundle;
7. import android.util.Log;
8. import android.view.View;
9. import android.widget.Button;
10. import android.widget.LinearLayout;
11. import android.widget.Toast;
12. public class testSQLite extends Activity {
13.     GVTable table;
14.     Button btnCreateDB, btnInsert, btnClose;
15.     SQLiteDatabase db;
16.     int id;//添加记录时的 id 累加标记, 必须全局
17.     private static final String TABLE_NAME = "stu";
18.     private static final String ID = "id";
19.     private static final String NAME = "name";
20.     private static final String PHONE = "phone";
21.     private static final String ADDRESS = "address";
22.     private static final String AGE = "age";
23.
24.     @Override
25.     public void onCreate(Bundle savedInstanceState) {
26.         super.onCreate(savedInstanceState);
27.         setContentView(R.layout.main);
28.         btnCreateDB = (Button) this.findViewById(R.id.btnCreateDB);
29.         btnCreateDB.setOnClickListener(new ClickEvent());
30.         btnInsert = (Button) this.findViewById(R.id.btnInsertRec);
31.         btnInsert.setOnClickListener(new ClickEvent());
32.         btnClose = (Button) this.findViewById(R.id.btnClose);

```

```

33.         btnClose.setOnClickListener(new ClickEvent());
34.         table=new GVTable(this);
35.         table.gvSetTableRowCount(8);//设置每个分页的 ROW 总数
36.         LinearLayout ly = (LinearLayout) findViewById(R.id.MainLinearLayout)
            ;
37.         table.setTableOnClickListener(new GVTable.OnTableClickListener() {
38.             @Override
39.             public void onTableClickListener(int x,int y,Cursor c) {
40.                 c.moveToPosition(y);
41.                 String str=c.getString(x)+" 位
置:("+String.valueOf(x)+","+String.valueOf(y)+")";
42.                 Toast.makeText(testSQLite.this, str, 1000).show();
43.             }
44.         });
45.         table.setOnPageSwitchListener(new GVTable.OnPageSwitchListener() {
46.
47.             @Override
48.             public void onPageSwitchListener(int pageID,int pageCount) {
49.                 String str="共有"+String.valueOf(pageCount)+
50.                 " 当前第"+String.valueOf(pageID)+"页";
51.                 Toast.makeText(testSQLite.this, str, 1000).show();
52.             }
53.         });
54.
55.         ly.addView(table);
56.     }
57.     class ClickEvent implements View.OnClickListener {
58.         @Override
59.         public void onClick(View v) {
60.             if (v == btnCreateDB) {
61.                 CreateDB();
62.             } else if (v == btnInsert) {
63.                 InsertRecord(16);//插入 16 条记录
64.                 table.gvUpdatePageBar("select count(*) from " + TABLE_NAME,d
b);
65.                 table.gvReadyTable("select * from " + TABLE_NAME,db);
66.             }else if (v == btnClose) {
67.                 table.gvRemoveAll();
68.                 db.close();
69.
70.             }
71.         }
72.     }
73.

```

```

74.  /**
75.   * 在内存创建数据库和数据表
76.   */
77.  void CreateDB() {
78.      // 在内存创建数据库
79.      db = SQLiteDatabase.create(null);
80.      Log.e("DB Path", db.getPath());
81.      String amount = String.valueOf(databaseList().length);
82.      Log.e("DB amount", amount);
83.      // 创建数据表
84.      String sql = "CREATE TABLE " + TABLE_NAME + " (" +
85.                  ID + " text not null, " + NAME + " text not null," +
86.                  ADDRESS + " text not null, " + PHONE + " text not null," +
87.                  AGE + " text not null "+");";
88.      try {
89.          db.execSQL("DROP TABLE IF EXISTS " + TABLE_NAME);
90.          db.execSQL(sql);
91.      } catch (SQLException e) {}
92.  }
93.  /**
94.   * 插入 N 条数据
95.   */
96.  void InsertRecord(int n) {
97.      int total = id + n;
98.      for (; id < total; id++) {
99.          String sql = "insert into " + TABLE_NAME + " (" +
100.                     ID + ", " + NAME + ", " + ADDRESS + ", " + PHONE + ", " + AGE
101.                     + ") values('" + String.valueOf(id) + "', 'man', 'address', '123456789', '18');"
102.          try {
103.              db.execSQL(sql);
104.          } catch (SQLException e) {}
105.      }
106.  }
107.  }
108.
109.
110. }

```

分页表格控件 **GVTable.java** 的源码:

[view plain](#)[copy](#) to clipboardprint?

```

1. package com.testSQLite;

```

```

2. import java.util.ArrayList;
3. import java.util.HashMap;
4. import android.content.Context;
5. import android.database.Cursor;
6. import android.database.sqlite.SQLiteDatabase;
7. import android.view.View;
8. import android.widget.AdapterView;
9. import android.widget.GridView;
10. import android.widget.LinearLayout;
11. import android.widget.SimpleAdapter;
12. import android.widget.AdapterView.OnItemClickListener;
13. public class GVTable extends LinearLayout {
14.     protected GridView gvTable,gvPage;
15.     protected SimpleAdapter saPageID,saTable;// 适配器
16.     protected ArrayList<HashMap<String, String>> srcPageID,srcTable;// 数据源
17.
18.     protected int TableRowCount=10;//分页时, 每页的 Row 总数
19.     protected int TableColCount=0;//每页 col 的数量
20.     protected SQLiteDatabase db;
21.     protected String rawSQL="";
22.     protected Cursor curTable;//分页时使用的 Cursor
23.     protected OnTableClickListener clickListener;//整个分页控件被点击时的回调函数
24.     protected OnPageSwitchListener switchListener;//分页切换时的回调函数
25.
26.     public GVTable(Context context) {
27.         super(context);
28.         this.setOrientation(VERTICAL);//垂直
29.         //-----
30.         gvTable=new GridView(context);
31.         addView(gvTable, new LinearLayout.LayoutParams(LayoutParams.FILL_PARENT,
32.             LayoutParams.WRAP_CONTENT));//宽长式样
33.
34.         srcTable = new ArrayList<HashMap<String, String>>();
35.         saTable = new SimpleAdapter(context,
36.             srcTable,// 数据来源
37.             R.layout.items,//XML 实现
38.             new String[] { "ItemText" },// 动态数组与 ImageItem对应的子项
39.             new int[] { R.id.ItemText }));
40.         // 添加并且显示
41.         gvTable.setAdapter(saTable);
42.         gvTable.setOnItemClickListener(new OnItemClickListener(){

```

```

43.         @Override
44.         public void onItemClick(AdapterView<?> arg0, View arg1, int arg2
45.             ,
46.             long arg3) {
47.             int y=arg2/curTable.getColumnCount()-1;//标题栏的不算
48.             int x=arg2 % curTable.getColumnCount();
49.             if (clickListener != null//分页数据被点击
50.                 && y!=-1) { //点中的不是标题栏时
51.                 clickListener.onTableClickListener(x,y,curTable);
52.             }
53.         }
54.     });
55.     //-----
56.     gvPage=new GridView(context);
57.     gvPage.setColumnWidth(40);//设置每个分页按钮的宽度
58.     gvPage.setNumColumns(GridView.AUTO_FIT);//分页按钮数量自动设置
59.     addView(gvPage, new LinearLayout.LayoutParams(LayoutParams.FILL_PARE
60.         NT,
61.         LayoutParams.WRAP_CONTENT)); //宽长式样
62.     srcPageID = new ArrayList<HashMap<String, String>>();
63.     saPageID = new SimpleAdapter(context,
64.         srcPageID, // 数据来源
65.         R.layout.items, //XML 实现
66.         new String[] { "ItemText" }, // 动态数组与 ImageItem 对应的子项
67.         new int[] { R.id.ItemText });
68.     // 添加并且显示
69.     gvPage.setAdapter(saPageID);
70.     // 添加消息处理
71.     gvPage.setOnItemClickListener(new OnItemClickListener(){
72.         @Override
73.         public void onItemClick(AdapterView<?> arg0, View arg1, int arg2
74.             ,
75.             long arg3) {
76.             LoadTable(arg2);//根据所选分页读取对应的数据
77.             if(switchListener!=null){ //分页切换时
78.                 switchListener.onPageSwitchListener(arg2,srcPageID.size(
79.                     ));
80.             }
81.         }
82.     });
83. }
84. /**
85.  * 清除所有数据

```

```

83.      */
84.      public void gvRemoveAll()
85.      {
86.          if(this.curTable!=null)
87.              curTable.close();
88.          srcTable.clear();
89.          saTable.notifyDataSetChanged();
90.
91.          srcPageID.clear();
92.          saPageID.notifyDataSetChanged();
93.
94.      }
95.      /**
96.       * 读取指定 ID 的分页数据,返回当前页的总数据
97.       * SQL:Select * From TABLE_NAME Limit 9 Offset 10;
98.       * 表示从 TABLE_NAME 表获取数据,跳过 10 行,取 9 行
99.       * @param pageID 指定的分页 ID
100.      */
101.      protected void LoadTable(int pageID)
102.      {
103.          if(curTable!=null)//释放上次的数据
104.              curTable.close();
105.
106.          String sql= rawSQL+" Limit "+String.valueOf(TableRowCount)+ " Offse
107.          t " +String.valueOf(pageID*TableRowCount);
108.          curTable = db.rawQuery(sql, null);
109.          gvTable.setNumColumns(curTable.getColumnCount());//表现为表格的关键
110.          点!
111.          TableColCount=curTable.getColumnCount();
112.          srcTable.clear();
113.          // 取得字段名称
114.          int colCount = curTable.getColumnCount();
115.          for (int i = 0; i < colCount; i++) {
116.              HashMap<String, String> map = new HashMap<String, String>();
117.              map.put("ItemText", curTable.getColumnName(i));
118.              srcTable.add(map);
119.          }
120.          // 列举出所有数据
121.          int recCount=curTable.getCount();
122.          for (int i = 0; i < recCount; i++) { //定位到一条数据
123.              curTable.moveToPosition(i);
124.              for(int ii=0;ii<colCount;ii++)//定位到一条数据中的每个字段

```



```

125.         {
126.             HashMap<String, String> map = new HashMap<String, String>()
127.             ;
128.             map.put("ItemText", curTable.getString(ii));
129.             srcTable.add(map);
130.         }
131.     }
132.     saTable.notifyDataSetChanged();
133. }
134. /**
135.  * 设置表格的最多显示的行数
136.  * @param row 表格的行数
137.  */
138. public void gvSetTableRowCount(int row)
139. {
140.     TableRowCount=row;
141. }
142.
143. /**
144.  * 取得表格的最大行数
145.  * @return 行数
146.  */
147. public int gvGetTableRowCount()
148. {
149.     return TableRowCount;
150. }
151.
152. /**
153.  * 取得当前分页的 Cursor
154.  * @return 当前分页的 Cursor
155.  */
156. public Cursor gvGetCurrentTable()
157. {
158.     return curTable;
159. }
160.
161. /**
162.  * 准备分页显示数据
163.  * @param rawSQL sql 语句
164.  * @param db 数据库
165.  */
166. public void gvReadyTable(String rawSQL, SQLiteDatabase db)
167. {

```

```

168.         this.rawQuery=rawSQL;
169.         this.db=db;
170.     }
171.
172.     /**
173.      * 刷新分页栏，更新按钮数量
174.      * @param sql SQL 语句
175.      * @param db 数据库
176.      */
177.     public void gvUpdatePageBar(String sql, SQLiteDatabase db)
178.     {
179.         Cursor rec = db.rawQuery(sql, null);
180.         rec.moveToLast();
181.         long recSize=rec.getLong(0);//取得总数
182.         rec.close();
183.         int pageNum=(int)(recSize/TableRowCount) + 1;//取得分页数
184.
185.         srcPageID.clear();
186.         for (int i = 0; i < pageNum; i++) {
187.             HashMap<String, String> map = new HashMap<String, String>();
188.             map.put("ItemText", "No." + String.valueOf(i));// 添加图像资源的
ID
189.             srcPageID.add(map);
190.         }
191.         saPageID.notifyDataSetChanged();
192.     }
193.     //-----
194.     /**
195.      * 表格被点击时的回调函数
196.      */
197.     public void setTableOnClickListener(OnClickListener click) {
198.         this.clickListener = click;
199.     }
200.
201.     public interface OnClickListener {
202.         public void onClickListener(int x,int y,Cursor c);
203.     }
204.     //-----
205.     /**
206.      * 分页栏被点击时的回调函数
207.      */
208.     public void setOnPageSwitchListener(OnPageSwitchListener pageSwitch) {
209.         this.switchListener = pageSwitch;

```

```
210.     }
211.     public interface OnPageSwitchListener {
212.         public void onPageSwitchListener(int pageID,int pageCount);
213.     }
214. }
```



本文档来自 **安卓巴士** ([www.apkbus.com](http://www.apkbus.com)) 整理总结

## 作者简介:

张国威 (用户名: hellogv)

<http://hi.csdn.net/hellogv>

- 性别: 男
- 生日: 1985 年 11 月 20 日
- 婚恋: 单身
- 居住: 江苏 南京
- 家乡: 广东 湛江
- MSN: hellogv@qq.com
- Email: hellogv@QQ.com

## Android提高第十篇之AudioRecord实现"助听器"

Android 可以通过 **MediaRecorder** 和 **AudioRecord** 这两个工具来实现录音, **MediaRecorder** 直接把麦克风的数据存到文件, 并且能够直接进行编码(如 **AMR**,**MP3** 等), 而 **AudioRecord** 则是读取麦克风的音频流。本文使用 **AudioRecord** 读取音频流, 使用 **AudioTrack** 播放音频流, 通过"边读边播放"以及增大音量的方式来实现一个简单的助听器程序。

**PS:** 由于目前的 **Android** 模拟器还不支持 **AudioRecord**, 因此本程序需要编译之后放到真机运行。

先贴出本文程序运行截图：

**PS:** 程序音量调节只是程序内部调节音量而已，要调到最大音量还需要手动设置系统音量。



使用 **AudioRecord** 必须要申请许可，在 **AndroidManifest.xml** 里面添加这句：

[view plaincopy to clipboardprint?](#)

```
1. <uses-permission android:name="android.permission.RECORD_AUDIO"></uses-permission>
```

**main.xml** 的源码如下：

[view plaincopy to clipboardprint?](#)

```
1. <?xml version="1.0" encoding="utf-8"?>
2. <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
3.     android:orientation="vertical" android:layout_width="fill_parent"
4.     android:layout_height="fill_parent">
5.
```

```

6.      <Button android:layout_height="wrap_content" android:id="@+id/btnRecord"
7.          android:layout_width="fill_parent" android:text="开始边录边放
"></Button>
8.      <Button android:layout_height="wrap_content"
9.          android:layout_width="fill_parent" android:text="停止
" android:id="@+id/btnStop"></Button>
10.     <Button android:layout_height="wrap_content" android:id="@+id/btnExit"
11.         android:layout_width="fill_parent" android:text="退出"></Button>
12.     <TextView android:id="@+id/TextView01" android:layout_height="wrap_conte
nt"
13.         android:text="程序音量调节
" android:layout_width="fill_parent"></TextView>
14.     <SeekBar android:layout_height="wrap_content" android:id="@+id/skbVolume
"
15.         android:layout_width="fill_parent"></SeekBar>
16.
17. </LinearLayout>

```

testRecord.java 的源码如下:

[view plaincopy to clipboardprint?](#)

```

1. package com.testRecord;
2.
3. import android.app.Activity;
4. import android.media.AudioFormat;
5. import android.media.AudioManager;
6. import android.media.AudioRecord;
7. import android.media.AudioTrack;
8. import android.media.MediaRecorder;
9. import android.os.Bundle;
10. import android.view.View;
11. import android.widget.Button;
12. import android.widget.SeekBar;
13. import android.widget.Toast;
14.
15. public class testRecord extends Activity {
16.     /** Called when the activity is first created. */
17.     Button btnRecord, btnStop, btnExit;
18.     SeekBar skbVolume;//调节音量
19.     boolean isRecording = false;//是否录放的标记
20.     static final int frequency = 44100;

```

```

21.     static final int channelConfiguration = AudioFormat.CHANNEL_CONFIGURATIO
N_MONO;
22.     static final int audioEncoding = AudioFormat.ENCODING_PCM_16BIT;
23.     int recBufSize,playBufSize;
24.     AudioRecord audioRecord;
25.     AudioTrack audioTrack;
26.
27.     @Override
28.     public void onCreate(Bundle savedInstanceState) {
29.         super.onCreate(savedInstanceState);
30.         setContentView(R.layout.main);
31.         setTitle("助听器");
32.         recBufSize = AudioRecord.getMinBufferSize(frequency,
33.             channelConfiguration, audioEncoding);
34.
35.         playBufSize=AudioTrack.getMinBufferSize(frequency,
36.             channelConfiguration, audioEncoding);
37.         // -----
38.         audioRecord = new AudioRecord(MediaRecorder.AudioSource.MIC, frequen
cy,
39.             channelConfiguration, audioEncoding, recBufSize);
40.
41.         audioTrack = new AudioTrack(AudioManager.STREAM_MUSIC, frequency,
42.             channelConfiguration, audioEncoding,
43.             playBufSize, AudioTrack.MODE_STREAM);
44.         //-----
45.         btnRecord = (Button) this.findViewById(R.id.btnRecord);
46.         btnRecord.setOnClickListener(new ClickEvent());
47.         btnStop = (Button) this.findViewById(R.id.btnStop);
48.         btnStop.setOnClickListener(new ClickEvent());
49.         btnExit = (Button) this.findViewById(R.id.btnExit);
50.         btnExit.setOnClickListener(new ClickEvent());
51.         skbVolume=(SeekBar)this.findViewById(R.id.skbVolume);
52.         skbVolume.setMax(100);//音量调节的极限
53.         skbVolume.setProgress(70);//设置 seekbar 的位置值
54.         audioTrack.setStereoVolume(0.7f, 0.7f);//设置当前音量大小
55.         skbVolume.setOnSeekBarChangeListener(new SeekBar.OnSeekBarChangeList
ener() {
56.
57.             @Override
58.             public void onStopTrackingTouch(SeekBar seekBar) {
59.                 float vol=(float)(seekBar.getProgress())/(float)(seekBar.get
Max());
60.                 audioTrack.setStereoVolume(vol, vol);//设置音量

```

```

61.         }
62.
63.         @Override
64.         public void onStartTrackingTouch(SeekBar seekBar) {
65.             // TODO Auto-generated method stub
66.         }
67.
68.         @Override
69.         public void onProgressChanged(SeekBar seekBar, int progress,
70.             boolean fromUser) {
71.             // TODO Auto-generated method stub
72.         }
73.     });
74. }
75.
76. @Override
77. protected void onDestroy() {
78.     super.onDestroy();
79.     android.os.Process.killProcess(android.os.Process.myPid());
80. }
81.
82. class ClickEvent implements View.OnClickListener {
83.
84.     @Override
85.     public void onClick(View v) {
86.         if (v == btnRecord) {
87.             isRecording = true;
88.             new RecordPlayThread().start();// 开一条线程边录边放
89.         } else if (v == btnStop) {
90.             isRecording = false;
91.         } else if (v == btnExit) {
92.             isRecording = false;
93.             testRecord.this.finish();
94.         }
95.     }
96. }
97.
98. class RecordPlayThread extends Thread {
99.     public void run() {
100.        try {
101.            byte[] buffer = new byte[recBufSize];
102.            audioRecord.startRecording();//开始录制
103.            audioTrack.play();//开始播放
104.

```

```
105.         while (isRecording) {
106.             //从 MIC 保存数据到缓冲区
107.             int bufferReadResult = audioRecord.read(buffer, 0,
108.                 recBufSize);
109.
110.             byte[] tmpBuf = new byte[bufferReadResult];
111.             System.arraycopy(buffer, 0, tmpBuf, 0, bufferReadResult
112.         );
113.             //写入数据即播放
114.             audioTrack.write(tmpBuf, 0, tmpBuf.length);
115.         }
116.         audioTrack.stop();
117.         audioRecord.stop();
118.     } catch (Throwable t) {
119.         Toast.makeText(testRecord.this, t.getMessage(), 1000);
120.     }
121. };
122. }
```





本文档来自 **安卓巴士** ([www.apkbus.com](http://www.apkbus.com)) 整理总结

## 作者简介:

---

张国威 (用户名: hellogv)

<http://hi.csdn.net/hellogv>

- 性别: 男
- 生日: 1985 年 11 月 20 日
- 婚恋: 单身
- 居住: 江苏 南京
- 家乡: 广东 湛江
- MSN: hellogv@qq.com
- Email: hellogv@QQ.com

## Android提高第十一篇之模拟信号示波器

---

上次简单地介绍了 [AudioRecord](#) 和 [AudioTrack](#) 的使用, 这次就结合 [SurfaceView](#) 实现一个 Android 版的手机模拟信号示波器(PS: 以前也讲过 [J2ME](#) 版的手机示波器)。最近物联网炒得很火, 作为手机软件开发者, 如何在不修改手机硬件电路的前提下实现与第三方传感器结合呢? 麦克风就是一个很好的 **ADC** 接口, 通过麦克风与第三方传感器结合, 再在软件里对模拟信号做相应的处理, 就可以提供更丰富的传感化应用。

先来看看本文程序运行的效果图(屏幕录像速度较慢, 真机实际运行起来会更加流畅):



本文程序使用 8000hz 的采样率，对 X 轴方向绘图的实时性要求较高，如果不降低 X 轴的分辨率，程序的实时性较差，因此程序对 X 轴数据缩小区间为 8 倍~16 倍。由于采用 16 位采样，因此 Y 轴数据的高度相对于手机屏幕来说也偏大，程序也对 Y 轴数据做缩小，区间为 1 倍~10 倍。在 SurfaceView 的 onTouchListener 方法里加入了波形基线的位置调节，直接在 SurfaceView 控件上触摸即可控制整体波形偏上或偏下显示。

main.xml 源码如下：

view plaincopy to clipboardprint?

```

1. <?xml version="1.0" encoding="utf-8"?>
2. <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
3.     android:orientation="vertical" android:layout_width="fill_parent"
4.     android:layout_height="fill_parent">
5.     <LinearLayout android:id="@+id/LinearLayout01"
6.         android:layout_height="wrap_content" android:layout_width="fill_pare
7.         nt"
8.         android:orientation="horizontal">
9.         <Button android:layout_height="wrap_content" android:id="@+id/btnSta
10.            rt"
11.                android:text="开始" android:layout_width="80dip"></Button>
12.            <Button android:layout_height="wrap_content" android:text="停止"

```

```

11.         android:id="@+id/btnExit" android:layout_width="80dip"></Button>
12.     <ZoomControls android:layout_width="wrap_content"
13.         android:layout_height="wrap_content" android:id="@+id/zctlX"></ZoomControls>
14.     <ZoomControls android:layout_width="wrap_content"
15.         android:layout_height="wrap_content" android:id="@+id/zctlY"></ZoomControls>
16. </LinearLayout>
17. <SurfaceView android:id="@+id/SurfaceView01"
18.     android:layout_height="fill_parent" android:layout_width="fill_parent"></SurfaceView>
19. </LinearLayout>

```

**ClsOscilloscope.java** 是实现示波器的类库，包含 **AudioRecord** 操作线程和 **SurfaceView** 绘图线程的实现，两个线程同步操作，代码如下：

view plaincopy to clipboardprint?

```

1. package com.testOscilloscope;
2. import java.util.ArrayList;
3. import android.graphics.Canvas;
4. import android.graphics.Color;
5. import android.graphics.Paint;
6. import android.graphics.Rect;
7. import android.media.AudioRecord;
8. import android.view.SurfaceView;
9. public class ClsOscilloscope {
10.     private ArrayList<short[]> inBuf = new ArrayList<short[]>();
11.     private boolean isRecording = false; // 线程控制标记
12.     /**
13.      * X轴缩小的比例
14.      */
15.     public int rateX = 4;
16.     /**
17.      * Y轴缩小的比例
18.      */
19.     public int rateY = 4;
20.     /**
21.      * Y轴基线
22.      */
23.     public int baseLine = 0;
24.     /**

```

```

25.     * 初始化
26.     */
27.     public void initOscilloscope(int rateX, int rateY, int baseLine) {
28.         this.rateX = rateX;
29.         this.rateY = rateY;
30.         this.baseLine = baseLine;
31.     }
32.     /**
33.     * 开始
34.     *
35.     * @param recBufSize
36.     *         AudioRecord 的 MinBufferSize
37.     */
38.     public void Start(AudioRecord audioRecord, int recBufSize, SurfaceView s
        fv,
39.         Paint mPaint) {
40.         isRecording = true;
41.         new RecordThread(audioRecord, recBufSize).start();// 开始录制线程
42.         new DrawThread(sfv, mPaint).start();// 开始绘制线程
43.     }
44.     /**
45.     * 停止
46.     */
47.     public void Stop() {
48.         isRecording = false;
49.         inBuf.clear();// 清除
50.     }
51.     /**
52.     * 负责从 MIC 保存数据到 inBuf
53.     *
54.     * @author GV
55.     *
56.     */
57.     class RecordThread extends Thread {
58.         private int recBufSize;
59.         private AudioRecord audioRecord;
60.         public RecordThread(AudioRecord audioRecord, int recBufSize) {
61.             this.audioRecord = audioRecord;
62.             this.recBufSize = recBufSize;
63.         }
64.         public void run() {
65.             try {
66.                 short[] buffer = new short[recBufSize];
67.                 audioRecord.startRecording();// 开始录制

```

```

68.         while (isRecording) {
69.             // 从MIC 保存数据到缓冲区
70.             int bufferReadResult = audioRecord.read(buffer, 0,
71.                 recBufSize);
72.             short[] tmpBuf = new short[bufferReadResult / rateX];
73.             for (int i = 0, ii = 0; i < tmpBuf.length; i++, ii = i
74.                 * rateX) {
75.                 tmpBuf[i] = buffer[ii];
76.             }
77.             synchronized (inBuf) {//
78.                 inBuf.add(tmpBuf);// 添加数据
79.             }
80.         }
81.         audioRecord.stop();
82.     } catch (Throwable t) {
83.     }
84. }
85. };
86. /**
87.  * 负责绘制 inBuf 中的数据
88.  *
89.  * @author GV
90.  *
91.  */
92. class DrawThread extends Thread {
93.     private int oldX = 0;// 上次绘制的 X 坐标
94.     private int oldY = 0;// 上次绘制的 Y 坐标
95.     private SurfaceView sfv;// 画板
96.     private int X_index = 0;// 当前画图所在屏幕 X 轴的坐标
97.     private Paint mPaint;// 画笔
98.     public DrawThread(SurfaceView sfv, Paint mPaint) {
99.         this.sfv = sfv;
100.        this.mPaint = mPaint;
101.    }
102.    public void run() {
103.        while (isRecording) {
104.            ArrayList<short[]> buf = new ArrayList<short[]>();
105.            synchronized (inBuf) {
106.                if (inBuf.size() == 0)
107.                    continue;
108.                buf = (ArrayList<short[]>) inBuf.clone();// 保存
109.                inBuf.clear();// 清除
110.            }
111.            for (int i = 0; i < buf.size(); i++) {

```

```

112.         short[] tmpBuf = buf.get(i);
113.         SimpleDraw(X_index, tmpBuf, rateY, baseLine);// 把缓冲区
           数据画出来
114.         X_index = X_index + tmpBuf.length;
115.         if (X_index > sfv.getWidth()) {
116.             X_index = 0;
117.         }
118.     }
119. }
120. }
121. /**
122.  * 绘制指定区域
123.  *
124.  * @param start
125.  *      X 轴开始的位置(全屏)
126.  * @param buffer
127.  *      缓冲区
128.  * @param rate
129.  *      Y 轴数据缩小的比例
130.  * @param baseLine
131.  *      Y 轴基线
132.  */
133. void SimpleDraw(int start, short[] buffer, int rate, int baseLine)
    {
134.         if (start == 0)
135.             oldX = 0;
136.         Canvas canvas = sfv.getHolder().lockCanvas(
137.             new Rect(start, 0, start + buffer.length, sfv.getHeight
                ()));// 关键:获取画布
138.         canvas.drawColor(Color.BLACK);// 清除背景
139.         int y;
140.         for (int i = 0; i < buffer.length; i++) {// 有多少画多少
141.             int x = i + start;
142.             y = buffer[i] / rate + baseLine;// 调节缩小比例, 调节基准线
143.             canvas.drawLine(oldX, oldY, x, y, mPaint);
144.             oldX = x;
145.             oldY = y;
146.         }
147.         sfv.getHolder().unlockCanvasAndPost(canvas);// 解锁画布, 提交画好
           的图像
148.     }
149. }
150. }

```

testOscilloscope.java 是主程序，控制 UI 和 ClsOscilloscope，代码如下：

view plaincopy to clipboardprint?

```
1. package com.testOscilloscope;
2. import android.app.Activity;
3. import android.graphics.Color;
4. import android.graphics.Paint;
5. import android.media.AudioFormat;
6. import android.media.AudioRecord;
7. import android.media.MediaRecorder;
8. import android.os.Bundle;
9. import android.view.MotionEvent;
10. import android.view.SurfaceView;
11. import android.view.View;
12. import android.view.View.OnTouchListener;
13. import android.widget.Button;
14. import android.widget.ZoomControls;
15. public class testOscilloscope extends Activity {
16.     /** Called when the activity is first created. */
17.     Button btnStart, btnExit;
18.     SurfaceView sfv;
19.     ZoomControls zctlX, zctlY;
20.
21.     ClsOscilloscope clsOscilloscope = new ClsOscilloscope();
22.
23.     static final int frequency = 8000; // 分辨率
24.     static final int channelConfiguration = AudioFormat.CHANNEL_CONFIGURATION_MONO;
25.     static final int audioEncoding = AudioFormat.ENCODING_PCM_16BIT;
26.     static final int xMax = 16; // X 轴缩小比例最大值, X 轴数据量巨大, 容易产生刷新延时
27.     static final int xMin = 8; // X 轴缩小比例最小值
28.     static final int yMax = 10; // Y 轴缩小比例最大值
29.     static final int yMin = 1; // Y 轴缩小比例最小值
30.
31.     int recBufSize; // 录音最小 buffer 大小
32.     AudioRecord audioRecord;
33.     Paint mPaint;
34.     @Override
35.     public void onCreate(Bundle savedInstanceState) {
36.         super.onCreate(savedInstanceState);
```

```

37.         setContentView(R.layout.main);
38.         //录音组件
39.         recBufSize = AudioRecord.getMinBufferSize(frequency,
40.             channelConfiguration, audioEncoding);
41.         audioRecord = new AudioRecord(MediaRecorder.AudioSource.MIC, frequen
cy,
42.             channelConfiguration, audioEncoding, recBufSize);
43.         //按键
44.         btnStart = (Button) this.findViewById(R.id.btnStart);
45.         btnStart.setOnClickListener(new ClickEvent());
46.         btnExit = (Button) this.findViewById(R.id.btnExit);
47.         btnExit.setOnClickListener(new ClickEvent());
48.         //画板和画笔
49.         sfv = (SurfaceView) this.findViewById(R.id.SurfaceView01);
50.         sfv.setOnTouchListener(new TouchEvent());
51.         mPaint = new Paint();
52.         mPaint.setColor(Color.GREEN);// 画笔为绿色
53.         mPaint.setStrokeWidth(1);// 设置画笔粗细
54.         //示波器类库
55.         clsOscilloscope.initOscilloscope(xMax/2, yMax/2, sfv.getHeight()/2);

56.
57.         //缩放控件, X 轴的数据缩小的比率高些
58.         zctlX = (ZoomControls)this.findViewById(R.id.zctlX);
59.         zctlX.setOnZoomInClickListener(new View.OnClickListener() {
60.             @Override
61.             public void onClick(View v) {
62.                 if(clsOscilloscope.rateX>xMin)
63.                     clsOscilloscope.rateX--;
64.                 setTitle("X 轴缩小"+String.valueOf(clsOscilloscope.rateX)+"倍
"
65.                     +","+ "Y 轴缩小
"+String.valueOf(clsOscilloscope.rateY)+"倍");
66.             }
67.         });
68.         zctlX.setOnZoomOutClickListener(new View.OnClickListener() {
69.             @Override
70.             public void onClick(View v) {
71.                 if(clsOscilloscope.rateX<xMax)
72.                     clsOscilloscope.rateX++;
73.                 setTitle("X 轴缩小"+String.valueOf(clsOscilloscope.rateX)+"倍
"
74.                     +","+ "Y 轴缩小
"+String.valueOf(clsOscilloscope.rateY)+"倍");

```



```

75.         }
76.     });
77.     zctlY = (ZoomControls)this.findViewById(R.id.zctlY);
78.     zctlY.setOnZoomInClickListener(new View.OnClickListener() {
79.         @Override
80.         public void onClick(View v) {
81.             if(clsOscilloscope.rateY>yMin)
82.                 clsOscilloscope.rateY--;
83.             setTitle("X 轴缩小"+String.valueOf(clsOscilloscope.rateX)+"倍
84.                 "+","+ "Y 轴缩小
85.                 "+String.valueOf(clsOscilloscope.rateY)+"倍");
86.         }
87.     });
88.     zctlY.setOnZoomOutClickListener(new View.OnClickListener() {
89.         @Override
90.         public void onClick(View v) {
91.             if(clsOscilloscope.rateY<yMax)
92.                 clsOscilloscope.rateY++;
93.             setTitle("X 轴缩小"+String.valueOf(clsOscilloscope.rateX)+"倍
94.                 "+","+ "Y 轴缩小
95.                 "+String.valueOf(clsOscilloscope.rateY)+"倍");
96.         }
97.     });
98.     @Override
99.     protected void onDestroy() {
100.         super.onDestroy();
101.         android.os.Process.killProcess(android.os.Process.myPid());
102.     }
103.
104.     /**
105.      * 按键事件处理
106.      * @author GV
107.      *
108.      */
109.     class ClickEvent implements View.OnClickListener {
110.         @Override
111.         public void onClick(View v) {
112.             if (v == btnStart) {
113.                 clsOscilloscope.baseLine=sfv.getHeight()/2;
114.                 clsOscilloscope.Start(audioRecord,recBufSize,sfv,mPaint);

```

```

115.         } else if (v == btnExit) {
116.             clsOscilloscope.Stop();
117.         }
118.     }
119. }
120. /**
121.  * 触摸屏动态设置波形图基线
122.  * @author GV
123.  *
124.  */
125. class TouchEvent implements OnTouchListener{
126.     @Override
127.     public boolean onTouch(View v, MotionEvent event) {
128.         clsOscilloscope.baseLine=(int)event.getY();
129.         return true;
130.     }
131.
132. }
133. }

```



本文档来自 **安卓巴士** ([www.apkbus.com](http://www.apkbus.com)) 整理总结

## 作者简介:

张国威 (用户名: hellogv)

<http://hi.csdn.net/hellogv>

- 性别: 男
- 生日: 1985 年 11 月 20 日
- 婚恋: 单身
- 居住: 江苏 南京
- 家乡: 广东 湛江
- MSN: hellogv@qq.com
- Email: [hellogv@QQ.com](mailto:hellogv@QQ.com)

## Android提高第十二篇之蓝牙传感应用

上次介绍了[Android利用麦克风采集并显示模拟信号](#)，这种采集手段适用于无IO控制、单纯读取信号的情况。如果传感器本身需要包含控制电路（例如采集血氧信号需要红外和红外线交替发射），那么传感器本身就需要带一片主控IC，片内采集并输出数字信号了。**Android**手机如何在不改硬件电路的前提下与这类数字传感器交互呢？可选的通信方式就有**USB**和**蓝牙**，两种方式各有好处：**USB**方式可以给传感器供电，**蓝牙**方式要自备电源；**USB**接口标准不一，**蓝牙**普遍支持**SPP**协议。本文选择**蓝牙**方式做介绍，介绍**Android**的**蓝牙API**以及**蓝牙客户端**的用法。

在**Android 2.0**，官方终于发布了**蓝牙API**（**2.0** 以下系统的非官方

的蓝牙API可以参考这

里: <http://code.google.com/p/android-bluetooth/>)。Android手机

一般以客户端的角色主动连接SPP协议设备(接上蓝牙模块的数字传感器), 连接流程是:

- 1.使用registerReceiver注册BroadcastReceiver来获取蓝牙状态、搜索设备等消息;
- 2.使用BlueAdatper的搜索;
- 3.在BroadcastReceiver的onReceive()里取得搜索所得的蓝牙设备信息(如名称, MAC, RSSI);
- 4.通过设备的MAC地址来建立一个BluetoothDevice对象;
- 5.由 BluetoothDevice 衍生出 BluetoothSocket, 准备 SOCKET 来读写设备;
- 6.通过 BluetoothSocket 的 createRfcommSocketToServiceRecord() 方法来选择连接的协议/服务, 这里用的是 SPP(UUID:00001101-0000-1000-8000-00805F9B34FB);
- 7.Connect 之后(如果还没配对则系统自动提示), 使用 BluetoothSocket 的 getInputStream()和 getOutputStream()来读写蓝牙设备。

先来看看本文程序运行的效果图,所选的 SPP 协议设备是一款单导联心电图采集表:



本文的代码较多，可以到这里下

载：<http://www.pudn.com/downloads305/sourcecode/comm/android/detail1359043.html>

本文程序包含两个 **Activity**(**testBlueTooth** 和 **WaveDiagram**),  
**testBlueTooth** 是搜索建立蓝牙连接。**BluetoothAdapter**、  
**BluetoothDevice** 和 **BluetoothSocket** 的使用很简单,除了前三者提供的功能外,还可以通过给系统发送消息来控制、获取蓝牙信息,例如:

注册 **BroadcastReceiver**:

[view plaincopy to clipboardprint?](#)

```
1. IntentFilter intent = new IntentFilter();
2. intent.addAction(BluetoothDevice.ACTION_FOUND);// 用 BroadcastReceiver 来取得搜索结果
3. intent.addAction(BluetoothDevice.ACTION_BOND_STATE_CHANGED);
4. intent.addAction(BluetoothAdapter.ACTION_SCAN_MODE_CHANGED);
```

```
5. intent.addAction(BluetoothAdapter.ACTION_STATE_CHANGED);
6. registerReceiver(searchDevices, intent);
```

在 **BroadcastReceiver** 的 **onReceive()**枚举所有消息的内容:

[view plaincopy to clipboardprint?](#)

```
1. String action = intent.getAction();
2.         Bundle b = intent.getExtras();
3.         Object[] lstName = b.keySet().toArray();
4.
5.         // 显示所有收到的消息及其细节
6.         for (int i = 0; i < lstName.length; i++) {
7.             String keyName = lstName[i].toString();
8.             Log.e(keyName, String.valueOf(b.get(keyName)));
9.         }
```

在 **DDMS** 里面可以看到 **BluetoothDevice.ACTION\_FOUND** 返回的消息:

android.bluetooth.device.extra.DEVICE	C0:CB:38:DB:16:C6
android.bluetooth.device.extra.CLASS	6010c
android.bluetooth.device.extra.RSSI	-51
android.bluetooth.device.extra.NAME	GV-PC
BluetoothEventLoop.cpp	event_filter: Received signal org.bluez.Adapter:DeviceFound from
android.bluetooth.device.extra.DEVICE	00:12:37:A7:51:DB
android.bluetooth.device.extra.CLASS	1a0114
android.bluetooth.device.extra.RSSI	-63
android.bluetooth.device.extra.NAME	dopod838

程序另外一个 **Activity**~~~**WaveDiagram** 用于读取蓝牙数据并绘制波形图, 这里要注意一下 **JAVA** 的 **byte** 的取值范围是跟 **C/C++** 不一样的, **Android** 接收到的 **byte** 数据要做 "& 0xFF" 处理, 转为 **C/C++** 等值的数据。



本文档来自 **安卓巴士** ([www.apkbus.com](http://www.apkbus.com)) 整理总结

## 作者简介:

张国威 (用户名: hellogv)

<http://hi.csdn.net/hellogv>

- 性别: 男
- 生日: 1985 年 11 月 20 日
- 婚恋: 单身
- 居住: 江苏 南京
- 家乡: 广东 湛江
- MSN: hellogv@qq.com
- Email: hellogv@QQ.com

## Android提高第十三篇之探秘蓝牙隐藏API

上次讲解**Android**的**蓝牙基本用法**, 这次讲得深入些, 探讨下**蓝牙**方面的**隐藏API**。用过**Android**系统设置(**Setting**)的人都知道**蓝牙**搜索之后可以**建立配对**和**解除配对**, 但是这两项功能的函数没有在**SDK**中给出, 那么如何去使用这两项功能呢? 本文利用**JAVA**的反射机制去调用这两项功能对应的函数: **createBond**和**removeBond**, 具体的发掘和实现步骤如下:

1.使用 **Git** 工具下载 **platform/packages/apps/Settings.git**, 在 **Setting** 源码中查找关于**建立配对**和**解除配对**的 **API**, 知道这两个 **API** 的宿主(**BluetoothDevice**);

2.使用反射机制对 **BluetoothDevice** 枚举其所有方法和常量，看看是否存在：

view plaincopy to clipboardprint?

```
1. static public void printAllInform(Class clsShow) {
2.     try {
3.         // 取得所有方法
4.         Method[] hideMethod = clsShow.getMethods();
5.         int i = 0;
6.         for (; i < hideMethod.length; i++) {
7.             Log.e("method name", hideMethod[i].getName());
8.         }
9.         // 取得所有常量
10.        Field[] allFields = clsShow.getFields();
11.        for (i = 0; i < allFields.length; i++) {
12.            Log.e("Field name", allFields[i].getName());
13.        }
14.    } catch (SecurityException e) {
15.        // throw new RuntimeException(e.getMessage());
16.        e.printStackTrace();
17.    } catch (IllegalArgumentException e) {
18.        // throw new RuntimeException(e.getMessage());
19.        e.printStackTrace();
20.    } catch (Exception e) {
21.        // TODO Auto-generated catch block
22.        e.printStackTrace();
23.    }
24. }
```

结果如下：

```
11-29 09:19:12.012: method name(452): cancelBondProcess
11-29 09:19:12.020: method name(452): cancelPairingUserInput
11-29 09:19:12.020: method name(452): createBond
11-29 09:19:12.020: method name(452): createInsecureRfcommSocket
11-29 09:19:12.027: method name(452): createRfcommSocket
11-29 09:19:12.027: method name(452): createRfcommSocketToServiceRecord
11-29 09:19:12.027: method name(452): createScoSocket
```



11-29 09:19:12.027: method name(452): describeContents  
 11-29 09:19:12.035: method name(452): equals  
 11-29 09:19:12.035: method name(452): fetchUuidsWithSdp  
 11-29 09:19:12.035: method name(452): getAddress  
 11-29 09:19:12.035: method name(452): getBluetoothClass  
 11-29 09:19:12.043: method name(452): getBondState  
 11-29 09:19:12.043: method name(452): getName  
 11-29 09:19:12.043: method name(452): getServiceChannel  
 11-29 09:19:12.043: method name(452): getTrustState  
 11-29 09:19:12.043: method name(452): getUuids  
 11-29 09:19:12.043: method name(452): hashCode  
 11-29 09:19:12.043: method name(452): isBluetoothDock  
**11-29 09:19:12.043: method name(452): removeBond**  
 11-29 09:19:12.043: method name(452): setPairingConfirmation  
 11-29 09:19:12.043: method name(452): setPasskey  
 11-29 09:19:12.043: method name(452): setPin  
 11-29 09:19:12.043: method name(452): setTrust  
 11-29 09:19:12.043: method name(452): toString  
 11-29 09:19:12.043: method name(452): writeToParcel  
 11-29 09:19:12.043: method name(452): convertPinToBytes  
 11-29 09:19:12.043: method name(452): getClass  
 11-29 09:19:12.043: method name(452): notify  
 11-29 09:19:12.043: method name(452): notifyAll  
 11-29 09:19:12.043: method name(452): wait  
 11-29 09:19:12.051: method name(452): wait  
 11-29 09:19:12.051: method name(452): wait

### 3.如果枚举发现 API 存在(SDK 却隐藏), 则自己实现调用方法:

[view plaincopy to clipboardprint?](#)

```

1.  /**
2.   * 与设备配对 参考源码: platform/packages/apps/Settings.git
3.   * \Settings\src\com\android\settings\bluetooth\CachedBluetoothDevice.java
4.   */
5.  static public boolean createBond(Class btClass,BluetoothDevice btDevice) throws Exception {
6.      Method createBondMethod = btClass.getMethod("createBond");
7.      Boolean returnValue = (Boolean) createBondMethod.invoke(btDevice);
8.      return returnValue.booleanValue();
9.  }
10.
11. /**
12.  * 与设备解除配对 参考源码: platform/packages/apps/Settings.git

```

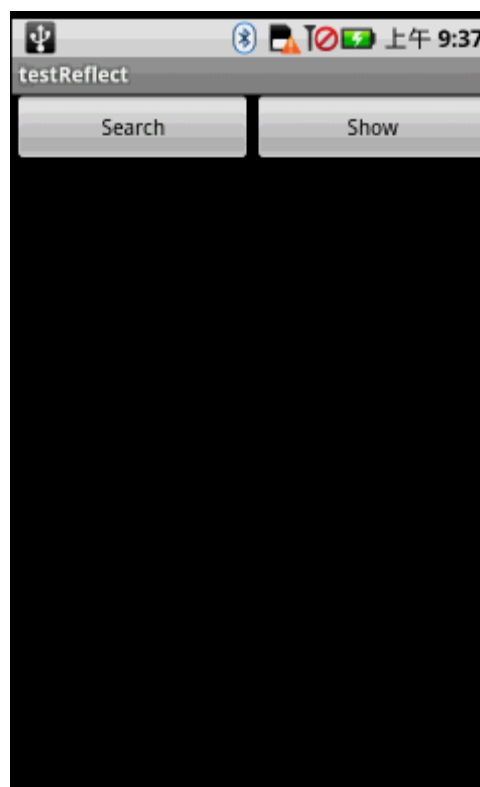
```

13. * \Settings\src\com\android\settings\bluetooth\CachedBluetoothDevice.java
14. */
15. static public boolean removeBond(Class btClass,BluetoothDevice btDevice) throws Exception {
16.     Method removeBondMethod = btClass.getMethod("removeBond");
17.     Boolean returnValue = (Boolean) removeBondMethod.invoke(btDevice);
18.     return returnValue.booleanValue();
19. }

```

PS:SDK 之所以不给出隐藏的 API 肯定有其原因,也许是出于安全性或者是后续版本兼容性的考虑,因此不能保证隐藏 API 能在所有 Android 平台上很好地运行。。。

本文程序运行效果如下:



main.xml 源码如下:

view plaincopy to clipboardprint?

```

1. <?xml version="1.0" encoding="utf-8"?>
2. <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"

```

```

3.     android:orientation="vertical" android:layout_width="fill_parent"
4.     android:layout_height="fill_parent">
5.     <LinearLayout android:id="@+id/LinearLayout01"
6.         android:layout_height="wrap_content" android:layout_width="fill_pare
7.         nt">
8.         <Button android:layout_height="wrap_content" android:id="@+id/btnSea
9.             rch"
10.             android:text="Search" android:layout_width="160dip"></Button>
11.         <Button android:layout_height="wrap_content"
12.             android:layout_width="160dip" android:text="Show" android:id="@+
13.             id/btnShow"></Button>
14.     </LinearLayout>
15.     <LinearLayout android:id="@+id/LinearLayout02"
16.         android:layout_width="wrap_content" android:layout_height="wrap_cont
17.         ent"></LinearLayout>
18.     <ListView android:id="@+id/ListView01" android:layout_width="fill_parent
19.         "
20.         android:layout_height="fill_parent">
21.     </ListView>
22. </LinearLayout>

```

工具类 `ClsUtils.java` 源码如下:

[view plain](#)[copy to clipboard](#)[print?](#)

```

1. package com.testReflect;
2.
3. import java.lang.reflect.Field;
4. import java.lang.reflect.Method;
5.
6. import android.bluetooth.BluetoothDevice;
7. import android.util.Log;
8.
9. public class ClsUtils {
10.
11.     /**
12.      * 与设备配对 参考源码: platform/packages/apps/Settings.git
13.      * \Settings\src\com\android\settings\bluetooth\CachedBluetoothDevice.java
14.      */
15.     static public boolean createBond(Class btClass,BluetoothDevice btDevice)
16.         throws Exception {
17.         Method createBondMethod = btClass.getMethod("createBond");
18.         Boolean returnValue = (Boolean) createBondMethod.invoke(btDevice);

```

```

18.         return returnValue.booleanValue();
19.     }
20.
21.     /**
22.      * 与设备解除配对 参考源码: platform/packages/apps/Settings.git
23.      * \Settings\src\com\android\settings\bluetooth\CachedBluetoothDevice.java
24.      */
25.     static public boolean removeBond(Class btClass,BluetoothDevice btDevice)
26.         throws Exception {
27.         Method removeBondMethod = btClass.getMethod("removeBond");
28.         Boolean returnValue = (Boolean) removeBondMethod.invoke(btDevice);
29.         return returnValue.booleanValue();
30.     }
31.
32.     *
33.     * @param clsShow
34.     */
35.     static public void printAllInform(Class clsShow) {
36.         try {
37.             // 取得所有方法
38.             Method[] hideMethod = clsShow.getMethods();
39.             int i = 0;
40.             for (; i < hideMethod.length; i++) {
41.                 Log.e("method name", hideMethod[i].getName());
42.             }
43.             // 取得所有常量
44.             Field[] allFields = clsShow.getFields();
45.             for (i = 0; i < allFields.length; i++) {
46.                 Log.e("Field name", allFields[i].getName());
47.             }
48.         } catch (SecurityException e) {
49.             // throw new RuntimeException(e.getMessage());
50.             e.printStackTrace();
51.         } catch (IllegalArgumentException e) {
52.             // throw new RuntimeException(e.getMessage());
53.             e.printStackTrace();
54.         } catch (Exception e) {
55.             // TODO Auto-generated catch block
56.             e.printStackTrace();
57.         }
58.     }
59. }

```

主程序 `testReflect.java` 的源码如下：

[view plain](#)[copy to clipboard](#)[print?](#)

```
1. package com.testReflect;
2.
3. import java.util.ArrayList;
4. import java.util.List;
5. import android.app.Activity;
6. import android.bluetooth.BluetoothAdapter;
7. import android.bluetooth.BluetoothDevice;
8. import android.content.BroadcastReceiver;
9. import android.content.Context;
10. import android.content.Intent;
11. import android.content.IntentFilter;
12. import android.os.Bundle;
13. import android.util.Log;
14. import android.view.View;
15. import android.widget.AdapterView;
16. import android.widget.AdapterView.OnItemClickListener;
17. import android.widget.Button;
18. import android.widget.ListView;
19. import android.widget.Toast;
20.
21. public class testReflect extends Activity {
22.     Button btnSearch, btnShow;
23.     ListView lvBTDevices;
24.     ArrayAdapter<String> adtDevices;
25.     List<String> lstDevices = new ArrayList<String>();
26.     BluetoothDevice btDevice;
27.     BluetoothAdapter btAdapt;
28.
29.     @Override
30.     public void onCreate(Bundle savedInstanceState) {
31.         super.onCreate(savedInstanceState);
32.         setContentView(R.layout.main);
33.
34.         btnSearch = (Button) this.findViewById(R.id.btnSearch);
35.         btnSearch.setOnClickListener(new ClickEvent());
36.         btnShow = (Button) this.findViewById(R.id.btnShow);
37.         btnShow.setOnClickListener(new ClickEvent());
38.
39.         lvBTDevices = (ListView) this.findViewById(R.id.ListView01);
40.         adtDevices = new ArrayAdapter<String>(testReflect.this,
```

```

41.         android.R.layout.simple_list_item_1, lstDevices);
42.     lvBTDevices.setAdapter(adapters);
43.     lvBTDevices.setOnItemClickListener(new ItemClickEvent());
44.
45.     btAdapt = BluetoothAdapter.getDefaultAdapter();// 初始化本机蓝牙功能
46.     if (btAdapt.getState() == BluetoothAdapter.STATE_OFF)// 开蓝牙
47.         btAdapt.enable();
48.
49.     // 注册 Receiver 来获取蓝牙设备相关的结果
50.     IntentFilter intent = new IntentFilter();
51.     intent.addAction(BluetoothDevice.ACTION_FOUND);
52.     intent.addAction(BluetoothDevice.ACTION_BOND_STATE_CHANGED);
53.     registerReceiver(searchDevices, intent);
54.
55. }
56.
57.
58. private BroadcastReceiver searchDevices = new BroadcastReceiver() {
59.     public void onReceive(Context context, Intent intent) {
60.         String action = intent.getAction();
61.         Bundle b = intent.getExtras();
62.         Object[] lstName = b.keySet().toArray();
63.
64.         // 显示所有收到的消息及其细节
65.         for (int i = 0; i < lstName.length; i++) {
66.             String keyName = lstName[i].toString();
67.             Log.e(keyName, String.valueOf(b.get(keyName)));
68.         }
69.         // 搜索设备时，取得设备的 MAC 地址
70.         if (BluetoothDevice.ACTION_FOUND.equals(action)) {
71.             BluetoothDevice device = intent
72.                 .getParcelableExtra(BluetoothDevice.EXTRA_DEVICE);
73.
74.             if (device.getBondState() == BluetoothDevice.BOND_NONE) {
75.                 String str = "未配对
|" + device.getName() + "|" + device.getAddress();
76.                 lstDevices.add(str); // 获取设备名称和 mac 地址
77.                 adapters.notifyDataSetChanged();
78.             }
79.         }
80.     }
81. };
82.
83. class ItemClickEvent implements AdapterView.OnItemClickListener {

```

```

84.
85.     @Override
86.     public void onItemClick(AdapterView<?> arg0, View arg1, int arg2,
87.         long arg3) {
88.         btAdapt.cancelDiscovery();
89.         String str = lstDevices.get(arg2);
90.         String[] values = str.split("\\|");
91.         String address=values[2];
92.
93.         btDevice = btAdapt.getRemoteDevice(address);
94.         try {
95.             if(values[0].equals("未配对"))
96.             {
97.                 Toast.makeText(testReflect.this, "由未配对转为已配对
98. ", 500).show();
99.                 ClsUtils.createBond(btDevice.getClass(), btDevice);
100.             }
101.             else if(values[0].equals("已配对"))
102.             {
103.                 Toast.makeText(testReflect.this, "由已配对转为未配对
104. ", 500).show();
105.                 ClsUtils.removeBond(btDevice.getClass(), btDevice);
106.             }
107.         } catch (Exception e) {
108.             // TODO Auto-generated catch block
109.             e.printStackTrace();
110.         }
111.     }
112.
113.     /**
114.     * 按键处理
115.     * @author GV
116.     *
117.     */
118.     class ClickEvent implements View.OnClickListener {
119.
120.         @Override
121.         public void onClick(View v) {
122.             if (v == btnSearch) { //搜索附近的蓝牙设备
123.                 lstDevices.clear();
124.

```

```

125.                Object[] lstDevice = btAdapt.getBondedDevices().toArray();
126.                for (int i = 0; i < lstDevice.length; i++) {
127.                    BluetoothDevice device=(BluetoothDevice)lstDevice[i];
128.                    String str = "已配对
|" + device.getName() + "|" + device.getAddress();
129.                    lstDevices.add(str); // 获取设备名称和 mac 地址
130.                    adtDevices.notifyDataSetChanged();
131.                }
132.                // 开始搜索
133.                setTitle("本机蓝牙地址: " + btAdapt.getAddress());
134.                btAdapt.startDiscovery();
135.            }
136.            else if(v==btnShow){//显示 BluetoothDevice 的所有方法和常量，包括隐
                藏 API
137.                ClsUtils.printAllInform(btDevice.getClass());
138.            }
139.
140.        }
141.
142.    }
143.
144.
145. }

```





本文档来自 **安卓巴士** ([www.apkbus.com](http://www.apkbus.com)) 整理总结

## 作者简介:

---

张国威 (用户名: hellogv)

<http://hi.csdn.net/hellogv>

- 性别: 男
- 生日: 1985 年 11 月 20 日
- 婚恋: 单身
- 居住: 江苏 南京
- 家乡: 广东 湛江
- MSN: hellogv@qq.com
- Email: hellogv@QQ.com

## Android提高第十四篇之探秘TelephonyManager

---

上次介绍了如何使用**JAVA**的反射机制来调用蓝牙的隐藏**API**, 这次继续练习**JAVA**的反射机制, 探秘**TelephonyManager**在**Framework**里包含却在**SDK**隐藏的几项功能。先来看看本文程序运行的效果图:



本文程序演示了以下功能：

- 1.所有来电自动接听；
- 2.所有来电自动挂断；
- 3.开启/关闭 Radio；
- 4.开启/关闭数据连接(WAP or NET 的连接)。

调用 `TelephonyManager` 的隐藏 API 是先参考 Framework 的 `\base\telephony\java\com\android\internal\telephony\ITelephony.aidl`，然后自己实现一个 `ITelephony.aidl`，最后在 `TelephonyManager` 中通过反射机制实例化自定义的 `ITelephony`，实例化之后就可以调用 `ITelephony` 里面的函数了。

本文程序需要在 `AndroidManifest.xml` 添加以下两行代码，以获得权限：

[view plaincopy to clipboardprint?](#)

```
1. <uses-permission android:name="android.permission.CALL_PHONE" />
2. <uses-permission android:name="android.permission.MODIFY_PHONE_STATE" />
```

main.xml 源码如下:

[view plaincopy to clipboardprint?](#)

```
1. <?xml version="1.0" encoding="utf-8"?>
2. <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
3.     android:orientation="vertical" android:layout_width="fill_parent"
4.     android:layout_height="fill_parent">
5.     <RadioGroup android:layout_height="wrap_content"
6.         android:layout_width="fill_parent" android:id="@+id/rGrpSelect">
7.         <RadioButton android:layout_height="wrap_content"
8.             android:layout_width="fill_parent" android:id="@+id/rbtnAutoAcce
9.             pt"
10.             android:text="所有来电自动接听"></RadioButton>
11.         <RadioButton android:layout_height="wrap_content"
12.             android:layout_width="fill_parent" android:id="@+id/rbtnAutoReje
13.             ct"
14.             android:text="所有来电自动挂断"></RadioButton>
15.     </RadioGroup>
16.     <ToggleButton android:layout_height="wrap_content"
17.         android:layout_width="fill_parent" android:id="@+id/tbtnRadioSwitch"
18.         android:textOn="Radio 已经启动" android:textOff="Radio 已经关闭"
19.         android:textSize="24dip" android:textStyle="normal"></ToggleButton>
20.     <ToggleButton android:layout_height="wrap_content"
21.         android:layout_width="fill_parent" android:id="@+id/tbtnDataConn"
22.         android:textSize="24dip" android:textStyle="normal" android:textOn="
    允许数据连接"
23.         android:textOff="禁止数据连接"></ToggleButton>
24. </LinearLayout>
```

PhoneUtils.java 是手机功能类，从 TelephonyManager 中实例化 ITelephony 并返回，源码如下:

[view plaincopy to clipboardprint?](#)

```
1. package com.testTelephony;
```

```

2.
3. import java.lang.reflect.Field;
4. import java.lang.reflect.Method;
5. import com.android.internal.telephony.ITelephony;
6. import android.telephony.TelephonyManager;
7. import android.util.Log;
8.
9. public class PhoneUtils {
10.     /**
11.      * 从 TelephonyManager 中实例化 ITelephony, 并返回
12.      */
13.     static public ITelephony getITelephony(TelephonyManager telMgr) throws E
        xception {
14.         Method getITelephonyMethod = telMgr.getClass().getDeclaredMethod("ge
            tITelephony");
15.         getITelephonyMethod.setAccessible(true); //私有化函数也能使用
16.         return (ITelephony) getITelephonyMethod.invoke(telMgr);
17.     }
18.
19.     static public void printAllInform(Class clsShow) {
20.         try {
21.             // 取得所有方法
22.             Method[] hideMethod = clsShow.getDeclaredMethods();
23.             int i = 0;
24.             for (; i < hideMethod.length; i++) {
25.                 Log.e("method name", hideMethod[i].getName());
26.             }
27.             // 取得所有常量
28.             Field[] allFields = clsShow.getFields();
29.             for (i = 0; i < allFields.length; i++) {
30.                 Log.e("Field name", allFields[i].getName());
31.             }
32.         } catch (SecurityException e) {
33.             // throw new RuntimeException(e.getMessage());
34.             e.printStackTrace();
35.         } catch (IllegalArgumentException e) {
36.             // throw new RuntimeException(e.getMessage());
37.             e.printStackTrace();
38.         } catch (Exception e) {
39.             // TODO Auto-generated catch block
40.             e.printStackTrace();
41.         }
42.     }
43. }

```

testTelephony.java 是主类,使用 PhoneStateListener 监听通话状态,以及实现上述 4 种电话控制功能, 源码如下:

[view plaincopy to clipboardprint?](#)

```
1. package com.testTelephony;
2.
3. import android.app.Activity;
4. import android.os.Bundle;
5. import android.telephony.PhoneStateListener;
6. import android.telephony.TelephonyManager;
7. import android.util.Log;
8. import android.view.View;
9. import android.widget.RadioGroup;
10. import android.widget.ToggleButton;
11.
12. public class testTelephony extends Activity {
13.     /** Called when the activity is first created. */
14.     RadioGroup rg;//来电操作单选框
15.     ToggleButton tbtnRadioSwitch;//Radio 开关
16.     ToggleButton tbtnDataConn;//数据连接的开关
17.     TelephonyManager telMgr;
18.     CallStateListener stateListener;
19.     int checkedId=0;
20.     @Override
21.     public void onCreate(Bundle savedInstanceState) {
22.         super.onCreate(savedInstanceState);
23.         setContentView(R.layout.main);
24.
25.         telMgr= (TelephonyManager) getSystemService(TELEPHONY_SERVICE);
26.         telMgr.listen(new CallStateListener(), CallStateListener.LISTEN_CALL
            _STATE);
27.
28.         PhoneUtils.printAllInform(TelephonyManager.class);
29.
30.         rg = (RadioGroup)findViewById(R.id.rGrpSelect);
31.         rg.setOnCheckedChangeListener(new CheckEvent());
32.         tbtnRadioSwitch=(ToggleButton)this.findViewById(R.id.tbtnRadioSwitch
            );
33.         tbtnRadioSwitch.setOnClickListener(new ClickEvent());
34.         try {
35.             tbtnRadioSwitch.setChecked(PhoneUtils.getITelephony(telMgr).isRa
                dioOn());
```

```

36.         } catch (Exception e) {
37.             Log.e("error",e.getMessage());
38.         }
39.         tbtnDataConn=(ToggleButton)this.findViewById(R.id.tbtnDataConn);
40.         tbtnDataConn.setOnClickListener(new ClickEvent());
41.         try {
42.             tbtnDataConn.setChecked(PhoneUtils.getITelephony(telMgr).isDataC
onnectivityPossible());
43.         } catch (Exception e) {
44.             Log.e("error",e.getMessage());
45.         }
46.     }
47.
48.     /**
49.      * 来电时的操作
50.      * @author GV
51.      *
52.      */
53.     public class CheckEvent implements RadioGroup.OnCheckedChangeListener{
54.
55.         @Override
56.         public void onCheckedChanged(RadioGroup group, int checkedId) {
57.             testTelephony.this.checkedId=checkedId;
58.         }
59.     }
60.
61.     /**
62.      * Radio 和数据连接的开关
63.      * @author GV
64.      *
65.      */
66.     public class ClickEvent implements View.OnClickListener{
67.
68.         @Override
69.         public void onClick(View v) {
70.             if (v == tbtnRadioSwitch) {
71.                 try {
72.                     PhoneUtils.getITelephony(telMgr).setRadio(tbtnRadioSwitc
h.isChecked());
73.                 } catch (Exception e) {
74.                     Log.e("error", e.getMessage());
75.                 }
76.             }
77.             else if(v==tbtnDataConn){

```

```

78.         try {
79.             if(tbtnDataConn.isChecked())
80.                 PhoneUtils.getITelephony(telMgr).enableDataConnectiv
ity();
81.             else if(!tbtnDataConn.isChecked())
82.                 PhoneUtils.getITelephony(telMgr).disableDataConnecti
vity();
83.         } catch (Exception e) {
84.             Log.e("error", e.getMessage());
85.         }
86.     }
87. }
88. }
89.
90. /**
91.  * 监视电话状态
92.  * @author GV
93.  *
94.  */
95. public class CallStateListener extends PhoneStateListener {
96.     @Override
97.     public void onCallStateChanged(int state, String incomingNumber) {
98.         if(state==TelephonyManager.CALL_STATE_IDLE)//挂断
99.         {
100.             Log.e("IDLE",incomingNumber);
101.         }
102.         else if(state==TelephonyManager.CALL_STATE_OFFHOOK)//接听
103.         {
104.             Log.e("OFFHOOK",incomingNumber);
105.         }
106.         else if(state==TelephonyManager.CALL_STATE_RINGING)//来电
107.         {
108.             if(testTelephony.this.checkedId==R.id.rbtnAutoAccept)
109.             {
110.                 try {
111.                     //需要
112.                     <uses-permission android:name="android.permission.MODIFY_PHONE_STATE" />
113.                     PhoneUtils.getITelephony(telMgr).silenceRinger();//
静铃
114.                     PhoneUtils.getITelephony(telMgr).answerRingingCall(
);//自动接听
115.                 } catch (Exception e) {
116.                     Log.e("error",e.getMessage());

```

```
117.         }
118.     }
119.     else if(testTelephony.this.checkedId==R.id.rbtnAutoReject)
120.     {
121.         try {
122.             PhoneUtils.getITelephony(telMgr).endCall();//挂断
123.             PhoneUtils.getITelephony(telMgr).cancelMissedCallsN
otification();//取消未接显示
124.         } catch (Exception e) {
125.             Log.e("error",e.getMessage());
126.         }
127.     }
128. }
129. super.onCallStateChanged(state, incomingNumber);
130. }
131. }
132. }
```





本文档来自 **安卓巴士** ([www.apkbus.com](http://www.apkbus.com)) 整理总结

作者简介:

---

张国威 (用户名: hellogv)

<http://hi.csdn.net/hellogv>

- 性别: 男
- 生日: 1985 年 11 月 20 日
- 婚恋: 单身
- 居住: 江苏 南京
- 家乡: 广东 湛江
- MSN: hellogv@qq.com
- Email: hellogv@QQ.com

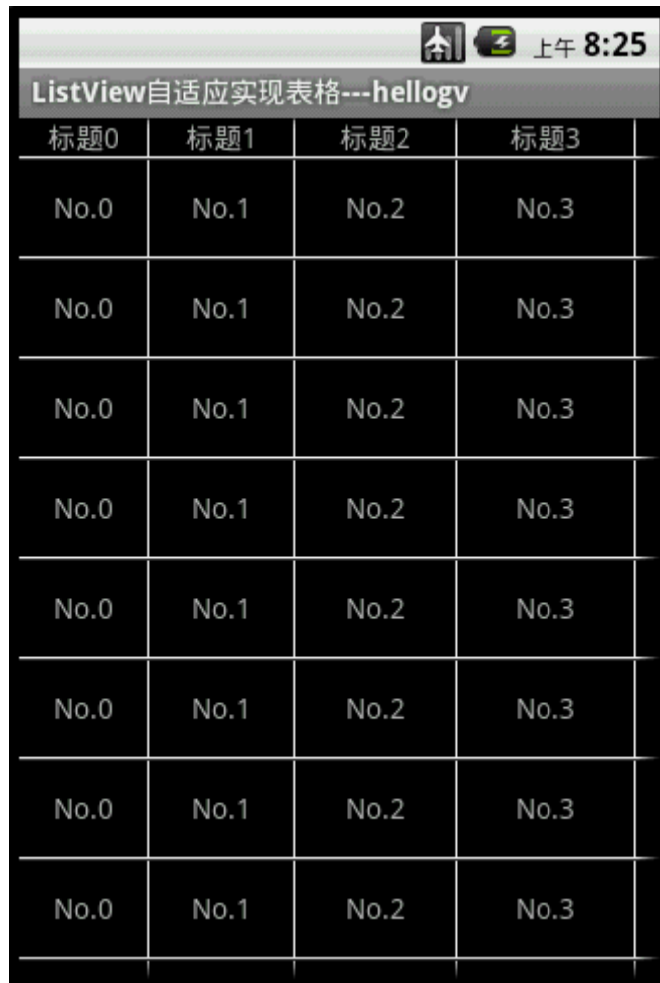
---

## Android提高第十五篇之ListView自适应实现表格

---

上次介绍了[使用GridView实现表格](#),这次就说说如何用**ListView**实现自适应的表格。**GridView**比**ListView**更容易实现自适应的表格,但是**GridView**每个格单元的大小固定,而**ListView**实现的表格可以自定义每个格单元的大小,但因此实现自适应表格也会复杂些(格单元大小不一)。另外,**GridView**实现的表格可以定位在具体某个格单元,而**ListView**实现的表格则只能定位在表格行。因此还是那句老话:根据具体的使用环境而选择**GridView** 或者 **ListView**实现表格。

先贴出本文程序运行的效果图:



标题0	标题1	标题2	标题3
No.0	No.1	No.2	No.3
No.0	No.1	No.2	No.3
No.0	No.1	No.2	No.3
No.0	No.1	No.2	No.3
No.0	No.1	No.2	No.3
No.0	No.1	No.2	No.3
No.0	No.1	No.2	No.3
No.0	No.1	No.2	No.3
No.0	No.1	No.2	No.3

本文实现的 **ListView** 表格, 可以每个格单元大小不一, 文本(**TextView**)或图片(**ImageView**)做格单元的数据, 不需要预先定义 **XML** 实现样式 (自适应的根本目标)。由于 **ListView** 置于 **HorizontalScrollView** 中, 因此对于列比较多/列数据比较长的数据表也能很好地适应其宽度。

**main.xml** 源码如下:

[view plaincopy to clipboardprint?](#)

```

1. <?xml version="1.0" encoding="utf-8"?>
2. <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
3.     android:orientation="vertical" android:layout_width="fill_parent"
4.     android:layout_height="fill_parent">
5.     <HorizontalScrollView android:id="@+id/HorizontalScrollView01"
6.         android:layout_height="fill_parent" android:layout_width="fill_paren
t">

```

```

7.         <ListView android:id="@+id/ListView01" android:layout_height="wrap_c
           ontent"
8.             android:layout_width="wrap_content"></ListView>
9.     </HorizontalScrollView>
10. </LinearLayout>

```

主类 `testMyListView.java` 的源码如下：

[view plaincopy to clipboardprint?](#)

```

1. package com.testMyListView;
2. import java.util.ArrayList;
3. import com.testMyListView.TableAdapter.TableCell;
4. import com.testMyListView.TableAdapter.TableRow;
5. import android.app.Activity;
6. import android.os.Bundle;
7. import android.view.View;
8. import android.widget.AdapterView;
9. import android.widget.ListView;
10. import android.widget.LinearLayout.LayoutParams;
11. import android.widget.Toast;
12. /**
13.  * @author hellogv
14.  */
15. public class testMyListView extends Activity {
16.     /** Called when the activity is first created. */
17.     ListView lv;
18.     @Override
19.     public void onCreate(Bundle savedInstanceState) {
20.         super.onCreate(savedInstanceState);
21.         setContentView(R.layout.main);
22.         this.setTitle("ListView 自适应实现表格---hellogv");
23.         lv = (ListView) this.findViewById(R.id.ListView01);
24.         ArrayList<TableRow> table = new ArrayList<TableRow>();
25.         TableCell[] titles = new TableCell[5]; // 每行 5 个单元
26.         int width = this.getWindowManager().getDefaultDisplay().getWidth()/t
           itles.length;
27.         // 定义标题
28.         for (int i = 0; i < titles.length; i++) {
29.             titles[i] = new TableCell("标题" + String.valueOf(i),
30.                                     width + 8 * i,
31.                                     LayoutParams.FILL_PARENT,
32.                                     TableCell.STRING);
33.         }

```

```

34.         table.add(new TableRow(titles));
35.         // 每行的数据
36.         TableCell[] cells = new TableCell[5]; // 每行 5 个单元
37.         for (int i = 0; i < cells.length - 1; i++) {
38.             cells[i] = new TableCell("No." + String.valueOf(i),
39.                                     titles[i].width,
40.                                     LayoutParams.FILL_PARENT,
41.                                     TableCell.STRING);
42.         }
43.         cells[cells.length - 1] = new TableCell(R.drawable.icon,
44.                                                  titles[cells.length - 1].wid
45.                                                  th,
46.                                                  LayoutParams.WRAP_CONTENT,
47.                                                  TableCell.IMAGE);
48.         // 把表格的行添加到表格
49.         for (int i = 0; i < 12; i++)
50.             table.add(new TableRow(cells));
51.         TableAdapter tableAdapter = new TableAdapter(this, table);
52.         lv.setAdapter(tableAdapter);
53.         lv.setOnItemClickListener(new ItemClickEvent());
54.         class ItemClickEvent implements AdapterView.OnItemClickListener {
55.             @Override
56.             public void onItemClick(AdapterView<?> arg0, View arg1, int arg2,
57.                                     long arg3) {
58.                 Toast.makeText(testMyListView.this, "选中第
59.                 "+String.valueOf(arg2)+"行", 500).show();
60.             }
61.         }

```

ListView 自适应实现 Table 的类 TableAdapter.java 代码如下:

PS: TableCell 是格单元的类, TableRow 是表格行的类, TableRowView

是实现表格行的组件。实现步骤: TableCell --> TableRow

(TableRowView) --> ListView

[view plaincopy to clipboardprint?](#)

```

1. package com.testMyListView;
2. import java.util.List;

```

```

3. import android.content.Context;
4. import android.graphics.Color;
5. import android.view.Gravity;
6. import android.view.View;
7. import android.view.ViewGroup;
8. import android.widget.BaseAdapter;
9. import android.widget.ImageView;
10. import android.widget.LinearLayout;
11. import android.widget.TextView;
12. public class TableAdapter extends BaseAdapter {
13.     private Context context;
14.     private List<TableRow> table;
15.     public TableAdapter(Context context, List<TableRow> table) {
16.         this.context = context;
17.         this.table = table;
18.     }
19.     @Override
20.     public int getCount() {
21.         return table.size();
22.     }
23.     @Override
24.     public long getItemId(int position) {
25.         return position;
26.     }
27.     public TableRow getItem(int position) {
28.         return table.get(position);
29.     }
30.     public View getView(int position, View convertView, ViewGroup parent) {
31.         TableRow tableRow = table.get(position);
32.         return new TableRowView(this.context, tableRow);
33.     }
34.     /**
35.      * TableRowView 实现表格行的样式
36.      * @author hellogv
37.      */
38.     class TableRowView extends LinearLayout {
39.         public TableRowView(Context context, TableRow tableRow) {
40.             super(context);
41.
42.             this.setOrientation(LinearLayout.HORIZONTAL);
43.             for (int i = 0; i < tableRow.getSize(); i++) { //逐个格单元添加到
行
44.                 TableCell tableCell = tableRow.getCellValue(i);

```

```

45.                LinearLayout.LayoutParams layoutParams = new LinearLayout.La
youtParams(
46.                    tableCell.width, tableCell.height);//按照格单元指定的
大小设置空间
47.                layoutParams.setMargins(0, 0, 1, 1);//预留空隙制造边框
48.                if (tableCell.type == TableCell.STRING) {//如果格单元是文本内
容
49.                    TextView textCell = new TextView(context);
50.                    textCell.setLines(1);
51.                    textCell.setGravity(Gravity.CENTER);
52.                    textCell.setBackgroundColor(Color.BLACK);//背景黑色
53.                    textCell.setText(String.valueOf(tableCell.value));
54.                    addView(textCell, layoutParams);
55.                } else if (tableCell.type == TableCell.IMAGE) {//如果格单元是
图像内容
56.                    ImageView imgCell = new ImageView(context);
57.                    imgCell.setBackgroundColor(Color.BLACK);//背景黑色
58.                    imgCell.setImageResource((Integer) tableCell.value);
59.                    addView(imgCell, layoutParams);
60.                }
61.            }
62.            this.setBackgroundColor(Color.WHITE);//背景白色，利用空隙来实现边
框
63.        }
64.    }
65.    /**
66.     * TableRow 实现表格的行
67.     * @author hellogv
68.     */
69.    static public class TableRow {
70.        private TableCell[] cell;
71.        public TableRow(TableCell[] cell) {
72.            this.cell = cell;
73.        }
74.        public int getSize() {
75.            return cell.length;
76.        }
77.        public TableCell getCellValue(int index) {
78.            if (index >= cell.length)
79.                return null;
80.            return cell[index];
81.        }
82.    }
83.    /**

```

```
84.      * TableCell 实现表格的格单元
85.      * @author hellogv
86.      */
87.      static public class TableCell {
88.          static public final int STRING = 0;
89.          static public final int IMAGE = 1;
90.          public Object value;
91.          public int width;
92.          public int height;
93.          private int type;
94.          public TableCell(Object value, int width, int height, int type) {
95.              this.value = value;
96.              this.width = width;
97.              this.height = height;
98.              this.type = type;
99.          }
100.     }
101. }
```



本文档来自 **安卓巴士** ([www.apkbus.com](http://www.apkbus.com)) 整理总结

## 作者简介:

张国威 (用户名: hellogv)

<http://hi.csdn.net/hellogv>

- 性别: 男
- 生日: 1985 年 11 月 20 日
- 婚恋: 单身
- 居住: 江苏 南京
- 家乡: 广东 湛江
- MSN: hellogv@qq.com
- Email: hellogv@QQ.com

## Android提高十六篇之使用NDK把彩图转换灰度图

在 **Android** 上使用 **JAVA** 实现彩图转换为灰度图, 跟 **J2ME** 上的实现类似, 不过遇到频繁地转换或者是大图转换时, 就必须使用 **NDK** 来提高速度了。本文主要通过 **JAVA** 和 **NDK** 这两种方式来分别实现彩图转换为灰度图, 并给出速度的对比。

先来简单地介绍一下 **Android** 的 **NDK** 使用步骤:

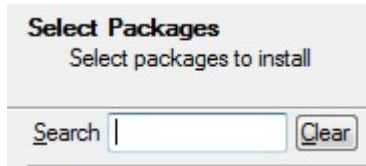
以**NDK r4** 为例, 或许以后新版的**NDK**的使用方法略有不同。

1、下载支持**C++**的**android-ndk-r4-crystax**, 支持**C++**的话可玩性更强.....

2、下载**cygwin**, 选择<ftp://mirrors.kernel.org>这个镜像, 搜索 **Devel**

**Install** 安装 **gcc** 和 **make** 等工具;





在搜索框里分别搜索 **gcc** 和 **make**，必须是 **Devel Install** 栏的。

3、Cygwin 安装目录下，找到 **home/username** 的目录下的 **.bash\_profile** 文件，打开文件在最后加上：

```
NDK=/cygdrive/d/android-ndk-r4-windows/android-ndk-r4
export NDK
```

PS：假设安装在 **d:\android-ndk-r4-windows\android-ndk-r4**。

4、运行 **cygwin**，通过 **cd** 命令去到 **NDK\samples\例子目录\**，运行 **\$NDK/ndk-build** 来编译该目录下的 **Android.mk**

以下是个人习惯.....

5、安装 **Eclipse** 的 **CDT**，官方下载 **cdt** 安装包，解压缩后把 **plugins** 和 **features** 复制覆盖到 **eclipse** 文件夹下即可

6、去到系统属性->环境变量->Path 添加 "**D:\cygwin\bin**"（假设 **cygwin** 安装在 **D:** 下），重启计算机，然后就可以在 **Eclipse** 里面建立基于 **cygwin** 的 **C/C++** 工程了，先通过这一步来验证 **NDK** 的程序能够编译成功，然后再通过第 4 步来生成 **SO** 文件。

接下来看看本文程序运行的效果：



从转换灰度图的耗时来说，NDK 的确比 JAVA 所用的时间短不少。

main.xml 源码如下：

[view plaincopy to clipboardprint?](#)

```
1. <?xml version="1.0" encoding="utf-8" ?>
2. - <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android" android:orientation="vertical" android:layout_width="fill_parent" android:layout_height="fill_parent">
3.   <Button android:layout_height="wrap_content" android:layout_width="fill_parent" android:id="@+id/btnJAVA" android:text="使用 JAVA 转换灰度图" />
4.   <Button android:layout_height="wrap_content" android:layout_width="fill_parent" android:id="@+id/btnNDK" android:text="使用 NDK 转换灰度图" />
5.   <ImageView android:id="@+id/ImageView01" android:layout_width="fill_parent" android:layout_height="fill_parent" />
6. </LinearLayout>
```

主程序 testToGray.java 的源码如下：

[view plaincopy to clipboardprint?](#)

```

1. package com.testToGray;
2.
3. import android.app.Activity;
4. import android.graphics.Bitmap;
5. import android.graphics.Bitmap.Config;
6. import android.graphics.drawable.BitmapDrawable;
7. import android.os.Bundle;
8. import android.view.View;
9. import android.widget.Button;
10. import android.widget.ImageView;
11.
12. public class testToGray extends Activity {
13.     /** Called when the activity is first created. */
14.     Button btnJAVA,btnNDK;
15.     ImageView imgView;
16.     @Override
17.     public void onCreate(Bundle savedInstanceState) {
18.         super.onCreate(savedInstanceState);
19.         setContentView(R.layout.main);
20.         this.setTitle("使用 NDK 转换灰度图---hellogy");
21.         btnJAVA=(Button)this.findViewById(R.id.btnJAVA);
22.         btnJAVA.setOnClickListener(new ClickEvent());
23.
24.         btnNDK=(Button)this.findViewById(R.id.btnNDK);
25.         btnNDK.setOnClickListener(new ClickEvent());
26.         imgView=(ImageView)this.findViewById(R.id.ImageView01);
27.     }
28.     class ClickEvent implements View.OnClickListener{
29.
30.         @Override
31.         public void onClick(View v) {
32.             if(v==btnJAVA)
33.             {
34.                 long current=System.currentTimeMillis();
35.                 Bitmap img=ConvertGrayImg(R.drawable.cat);
36.                 long performance=System.currentTimeMillis()-current;
37.                 //显示灰度图
38.                 imgView.setImageBitmap(img);
39.                 testToGray.this.setTitle("w:"+String.valueOf(img.getWidth())
40.                     +",h:"+String.valueOf(img.getHeight())
41.                     +" JAVA 耗时 "+String.valueOf(performance)+" 毫秒");
42.             }
43.             else if(v==btnNDK)
44.             {

```

```

44.         long current=System.currentTimeMillis();
45.
46.         //先打开图像并读取像素
47.         Bitmap img1=((BitmapDrawable) getResources().getDrawable(R.d
rawable.cat)).getBitmap();
48.         int w=img1.getWidth(),h=img1.getHeight();
49.         int[] pix = new int[w * h];
50.         img1.getPixels(pix, 0, w, 0, 0, w, h);
51.         //通过 ImgToGray.so 把彩色像素转为灰度像素
52.         int[] resultInt=LibFuns.ImgToGray(pix, w, h);
53.         Bitmap resultImg=Bitmap.createBitmap(w, h, Config.RGB_565);
54.         resultImg.setPixels(resultInt, 0, w, 0, 0,w, h);
55.         long performance=System.currentTimeMillis()-current;
56.         //显示灰度图
57.         imageView.setImageBitmap(resultImg);
58.         testToGray.this.setTitle("w:"+String.valueOf(img1.getWidth()
)+" ,h:"+String.valueOf(img1.getHeight())
59.             +" NDK 耗时 " +String.valueOf(performance)+" 毫秒");
60.     }
61. }
62. }
63.
64. /**
65.  * 把资源图片转为灰度图
66.  * @param resID 资源 ID
67.  * @return
68.  */
69. public Bitmap ConvertGrayImg(int resID)
70. {
71.     Bitmap img1=((BitmapDrawable) getResources().getDrawable(resID)).get
Bitmap();
72.
73.     int w=img1.getWidth(),h=img1.getHeight();
74.     int[] pix = new int[w * h];
75.     img1.getPixels(pix, 0, w, 0, 0, w, h);
76.
77.     int alpha=0xFF<<24;
78.     for (int i = 0; i < h; i++) {
79.         for (int j = 0; j < w; j++) {
80.             // 获得像素的颜色
81.             int color = pix[w * i + j];
82.             int red = ((color & 0x00FF0000) >> 16);
83.             int green = ((color & 0x0000FF00) >> 8);

```

```

84.         int blue = color & 0x000000FF;
85.         color = (red + green + blue)/3;
86.         color = alpha | (color << 16) | (color << 8) | color;
87.         pix[w * i + j] = color;
88.     }
89. }
90. Bitmap result=Bitmap.createBitmap(w, h, Config.RGB_565);
91. result.setPixels(pix, 0, w, 0, 0,w, h);
92. return result;
93. }
94. }

```

封装 NDK 函数的 JAVA 类 LibFuns.java 的源码如下:

[view plaincopy to clipboardprint?](#)

```

1. package com.testToGray;
2. public class LibFuns {
3.     static {
4.         System.loadLibrary("ImgToGray");
5.     }
6.
7.     /**
8.      * @param width the current view width
9.      * @param height the current view height
10.     */
11.
12.     public static native int[] ImgToGray(int[] buf, int w, int h);
13. }

```

彩图转换为灰度图的 ImgToGray.cpp 源码:

[view plaincopy to clipboardprint?](#)

```

1. #include <jni.h>
2. #include <stdio.h>
3. #include <stdlib.h>
4.
5. extern "C" {
6. JNIEXPORT jintArray JNICALL Java_com_testToGray_LibFuns_ImgToGray(
7.     JNIEnv* env, jobject obj, jintArray buf, int w, int h);
8. }
9. ;
10.

```

```

11. JNIEXPORT jintArray JNICALL Java_com_testToGray_LibFuns_ImgToGray(
12.     JNIEnv* env, jobject obj, jintArray buf, int w, int h) {
13.     jint *cbuf;
14.
15.     cbuf = env->GetIntArrayElements(buf, false);
16.     if (cbuf == NULL) {
17.         return 0; /* exception occurred */
18.     }
19.     int alpha = 0xFF << 24;
20.     for (int i = 0; i < h; i++) {
21.         for (int j = 0; j < w; j++) {
22.             // 获得像素的颜色
23.             int color = cbuf[w * i + j];
24.             int red = ((color & 0x00FF0000) >> 16);
25.             int green = ((color & 0x0000FF00) >> 8);
26.             int blue = color & 0x000000FF;
27.             color = (red + green + blue) / 3;
28.             color = alpha | (color << 16) | (color << 8) | color;
29.             cbuf[w * i + j] = color;
30.         }
31.     }
32.
33.     int size=w * h;
34.     jintArray result = env->NewIntArray(size);
35.     env->SetIntArrayRegion(result, 0, size, cbuf);
36.
37.     env->ReleaseIntArrayElements(buf, cbuf, 0);
38.
39.     return result;
40. }

```

## Android.mk 的源码:

[view plaincopy to clipboardprint?](#)

```

1. LOCAL_PATH:= $(call my-dir)
2.
3. include $(CLEAR_VARS)
4.
5. LOCAL_MODULE      := ImgToGray
6. LOCAL_SRC_FILES   := ImgToGray.cpp
7.
8. include $(BUILD_SHARED_LIBRARY)

```



本文档来自 **安卓巴士** ([www.apkbus.com](http://www.apkbus.com)) 整理总结

## 作者简介:

---

**张国威** (用户名: hellogv)

<http://hi.csdn.net/hellogv>

- 性别: 男
- 生日: 1985 年 11 月 20 日
- 婚恋: 单身
- 居住: 江苏 南京
- 家乡: 广东 湛江
- MSN: hellogv@qq.com
- Email: hellogv@QQ.com